

Sterling Selling and Fulfillment Foundation



Integration Guide

Release 9.1

Sterling Selling and Fulfillment Foundation



Integration Guide

Release 9.1

Note

Before using this information and the product it supports, read the information in "Notices" on page 151.

Copyright

This edition applies to the 9.1 Version of IBM Sterling Selling and Fulfillment Foundation and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 1999, 2011.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Integration Overview 1

Integrating with Sterling Selling and Fulfillment Foundation	1
Application Integration Architecture	1
Integration with Warehouse Management Systems	2
Integration with Parcel Carrier Adapters	2
Integration with the Loftware Print Server and Label Manager	2
Integration with Material Handling Equipment	3
Integration with Enterprise Resource Planning Systems	3
Integration with Point of Sale Systems	3
Integration with JMS Systems	3
Integration with Financial Systems	3
Rapid Deployment Features	3

Chapter 2. Integrating with the Distribution Center System 5

Introduction to Distribution Center System Integration	5
DCS Purchase Order Interface Function	5
Purchase Order Workflow	5
Understanding Purchase Order Transactions	7
Configuring Purchase Order Time-Triggered Transactions	9
Configuring the Purchase Order Pipeline	9
DCS Purchase Order Interface	10
DCS Shipment Interface	13
Understanding Order Transactions	13
Configuring DCS Shipment Time-Triggered Transactions	14
DCS Order Release Interface	15
DCS Shipment Confirmation	22
DCS Inventory Interface	26
DCS Inventory Upload	26
DCS Inventory Download	27
DCS Returns Interface	29
Return Order Integration Workflow	30
Determining the Enterprise Code for Blind Return During Upload	31
Configuring Return Order Integration with DCS	31
Return Order Interface Data Mapping	33
Assumptions and Limitations	38

Chapter 3. Integrating with Stand-Alone Sterling Warehouse Management System 41

Limitation on Use of Services with Stand-Alone Sterling Warehouse Management System	41
Installing Integration Pack for Receipt and Inventory Change Upload Interfaces on a Sterling Warehouse Management System Instance	41

Installing Integration Pack for Receipt and Inventory Change Upload Interfaces on a Sterling Distributed Order Management Instance	41
Uploading Receipts	42
Uploading Receipt Information	42
Uploading Receipt Adjustment Information	44
Loading Receipt Information from a Node	45
Loading Receipt Adjustment Information from a Node	47
Uploading Inventory Changes at a Node	48
Uploading Updated Inventory Information	48
Loading Inventory Information from a Node	50
Uploading Inventory Snapshots	51
Generating Inventory Snapshot Files	51

Chapter 4. Integrating with Third-Party Warehouse Management Systems 53

Introduction to Third-Party Warehouse Management System Integration	53
Third-Party Warehouse Management Systems	53
Third-Party Shipment Advice	53
Third-Party Inventory Change	53

Chapter 5. Integrating with the Loftware Print Server and Label Manager 55

Overview of Loftware Print Server and Label Manager Integration	55
Printing Standard Labels	56
Designing Custom Labels	57
Displaying Page Numbers	57
File Naming Conventions for Custom Labels	57
File Design Conventions for Labels	57
Creating a New Label Format	57
Defining Custom Print Services	61
Configuring a Print Pack List Service	61

Chapter 6. Integrating with Parcel Carrier Adapters 69

Overview of Parcel Carrier Adapter Integration	69
APIs Invoked During Parcel Carrier Adapter Integration	69
Field-Level Mapping Between the openManifest API on the Sterling Warehouse Management System and the openManifest API on the Carrier Adapter	70
Mappings Between the addContainerToManifest API on the Sterling Warehouse Management System and the shipCarton API on the Carrier Adapter	71
Mapping Between the removeContainerFromManifest API on the Sterling Warehouse Management System and the deleteCarton API on the Carrier Adapter	81
Mapping Between the closeManifest API on the Sterling Warehouse Management System and the closeManifest API on the Carrier Adapter	81

Integration Dependencies	82
------------------------------------	----

Chapter 7. Integrating with Material Handling Equipment 83

Overview of Material Handling Equipment Integration	83
Integrating with Pick-to-Light Systems	83
Integrating with Put-to-Light Systems	84
Integrating with Carousel or Automated Storage and Retrieval Systems	86
Integration When a Product is Being Put Away	86
Integration When a Product is Being Retrieved	86
Integration When a Product is Being Counted	87
Integrating with Automatic Guided Vehicles	88
Integrating with Inbound Sorters	89
Integrating with Pack Sorters	89
Integrating with Shipping Sorters	90
Integrating with Cube-a-Scans	91
Integrating with Weighing Scales	91
Integrating with Mettler Toledo Weighing Scales	91
Integrating with Other Weighing Scales	92

Chapter 8. Integrating with Enterprise Resource Planning Systems 95

Overview of Integration with ERP Components	95
Integration Data Flow Diagram	96
Integration Protocol	96
ERP Integration Specification: Order Management	97
Customer Download from an ERP System to the Sterling Warehouse Management System	97
Shipment/Order Release Download from an ERP System to the Sterling Warehouse Management System	97
Shipment Confirmation Upload from the Sterling Warehouse Management System to an ERP System	97
ERP Integration Specification: Purchasing	98
Vendor Download from an ERP System to the Sterling Warehouse Management System	98
Purchase Order Download from an ERP System to the Sterling Warehouse Management System	98
Purchase Order Closure Download from an ERP System to the Sterling Warehouse Management System	98
ASN Download from an ERP System to the Sterling Warehouse Management System	98
Receipt Upload from the Sterling Warehouse Management System to an ERP System	99
ERP Integration Specification: Inventory	99
Item Download from an ERP System to the Sterling Warehouse Management System	99
Item Attributes Upload from the Sterling Warehouse Management System to an ERP System	99
Inventory Change Upload from the Sterling Warehouse Management System to an ERP System	100
Inventory Snapshot Upload from the Sterling Warehouse Management System to an ERP System	100

ERP System Integration Specification: WIP	100
BOM Download from an ERP System to the Sterling Warehouse Management System	100
Work Order Download from an ERP System to the Sterling Warehouse Management System	101
Work Order Demand Upload for Manually Created Work Orders from the Sterling Warehouse Management System to ERP	101
Work Order Confirmation Upload from the Sterling Warehouse Management System to an ERP System	101
Close Work Order from the Sterling Warehouse Management System to an ERP System	102
ERP Integration Specification: Returns	102
Return Order Download from ERP to the Sterling Warehouse Management System	102
Return Order Closure Download from an ERP System to the Sterling Warehouse Management System	102
Receipt Upload from the Sterling Warehouse Management System to an ERP System	103

Chapter 9. Point of Sale System Integration 105

Integrating with Point of Sale Systems	105
API Invoked During Point of Sale Integration	105

Chapter 10. Integrating User and Item Data with External Systems 107

External System Integration Overview	107
Order Management Integration	107
APIs Invoked During Order Management Integration	107
User and Item Synchronization	108
Item Synchronization Services in Sterling Selling and Fulfillment Foundation	108
SendItemChanges Service	108
ReceiveItemChanges Service	109
Customer Synchronization Services in Sterling Selling and Fulfillment Foundation	110
SendCustomerChanges Service	110
ReceiveCustomerChanges Service	111
Modifying Customer Event Templates	112
Data Mapping	113
Customer Data Mapping	113
Item Data Mapping	114

Chapter 11. Integrating with JMS Systems 119

Introduction to Integrating with JMS	119
Configuring Oracle WebLogic JMS	119
WebLogic Time-Out Considerations for Transacted Sessions	120
Before You Begin Configuring IBM WebSphere MQ	121
Creating the Queue Manager and Queues for IBM WebSphere MQ	121
Configuring a Queue Manager to Client Connection for IBM WebSphere MQ	121
Configuring Sterling Selling and Fulfillment Foundation to Use WebSphere MQ Queues	123

Configuring IBM WebSphere MQ for Access by WebSphere's JNDI Namespace	123
Configuring IBM WebSphere Default Messaging	125
Configuring Sterling Selling and Fulfillment Foundation to Use WebSphere Default Messaging	125
JBoss Messaging JMS	126
Creating Queues in JBoss Messaging JMS	126
Configuring Sterling Selling and Fulfillment Foundation to Use JBoss Messaging Queues	127
Configuring TIBCO JMS	128
TIBCO JMS Attributes	128
Configuring TIBCO JMS as an Agent Queue	129
Configuring the Sterling Selling and Fulfillment Foundation To Use TIBCO Messaging Queues	130

Chapter 12. Integrating with Financial Systems 131

Requirements for Financial System Integration	131
Load Initial Inventory Cost Data	131
Configuring Process-Specific Events	131

Receipt Process	131
Sales Order Creation Process	132
Shipment Confirmation Process	132
Invoice Process	133
Work Order Confirmation Process	133
Inventory Adjustment Process	134
Return Order Process	134
Callback from Financial System for Inventory Value Adjustment	134

Chapter 13. Rapid Deployment

Features 137

Rapid Deployment Feature Overview	137
Interface Field Mapping Documents	137
Generating Interface Field Mapping Template Documents	137
Initial Data Loading	139
Initial Data-Loading Services	139

Notices 151

Chapter 1. Integration Overview

Integrating with Sterling Selling and Fulfillment Foundation

IBM® Sterling Selling and Fulfillment Foundation can be integrated with other IBM offerings, such as the Distributed Center Solution and third-party applications, through the services defined using the Service Definition Framework.

Sterling Selling and Fulfillment Foundation provides integration with:

- The Distribution Center System
- Third-Party Warehouse Management System
- The Parcel Carrier Adapters
- Loftware Print Server and Label Manager
- Material Handling Equipment
- Enterprise Resource Planning Systems
- Point of Sale Systems
- JMS Systems
- Financial Systems
- Interface Field Mapping Documents

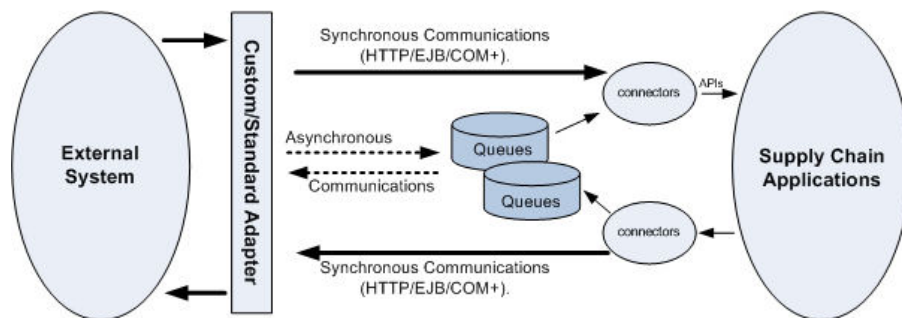
For more information about defining specific services, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*

Note: If you try to configure more than one action serially using the Service Definition Framework, the Applications Manager throws an error message, "A continue link must be attached to the next condition or action." To avoid this error, group these actions and replace them with one service.

Application Integration Architecture

Adapters connect to external systems through the Service Definition Framework for data transformation.

The following figure shows how the Service Definition Framework fits into the applications integration architecture of Sterling Selling and Fulfillment Foundation, the various adapters that perform data transformation, and the goals of the transformations.



For more information about the adapters used within Sterling Selling and Fulfillment Foundation, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Integration with Warehouse Management Systems

Sterling Selling and Fulfillment Foundation provides real-time integration with the IBM Sterling Warehouse Management System (Sterling Warehouse Management System).

Sterling Selling and Fulfillment Foundation supports integration with the Distribution Center System (DCS). The DCS application supports warehouse inventory, distribution, returns and activities. Typically, it is used in distribution centers for fulfilling large numbers of orders, with items required in quantities of a case or less. For information about integrating Sterling Selling and Fulfillment Foundation with the Distribution Center System, see “Introduction to Distribution Center System Integration” on page 5.

Sterling Selling and Fulfillment Foundation also supports integration with third-party warehouse management systems. For information about integrating Sterling Selling and Fulfillment Foundation with third-party warehouse management systems, see “Introduction to Third-Party Warehouse Management System Integration” on page 53.

Integration with Parcel Carrier Adapters

Sterling Selling and Fulfillment Foundation provides integration with the Parcel Carrier Adapters (Carrier Adapter), which manages all the carrier-integration related functions of Sterling Selling and Fulfillment Foundation. Sterling Selling and Fulfillment Foundation interfaces with the Carrier Adapter to use its carrier-integration functions.

The Carrier Adapter is regularly updated with the latest carrier data, such as rates and special services, and hence can act as a centralized carrier-integration database and business rules manager. The Carrier Adapter helps companies to quickly meet the changing requirements initiated by both carriers and customers, in the most efficient way.

The Carrier Adapter has a data driven design. The functionality is defined in terms of the relation between data elements stored in the database. Carriers having similar functionality can be incorporated into an installation with minimal engineering effort.

The Carrier Adapter is now integrated into Sterling Selling and Fulfillment Foundation. For more information about the Carrier Adapter and how to configure it, see the *Sterling Selling and Fulfillment Foundation: Parcel Carrier Adapter* .

Integration with the Loftware Print Server and Label Manager

Sterling Selling and Fulfillment Foundation provides integration with the Loftware Print Server and Loftware Label Manager for printing reports and designing custom labels. You can also design custom print services using the Service Definition Framework. For more information about the print server and label manager, see “Overview of Loftware Print Server and Label Manager Integration” on page 55.

Integration with Material Handling Equipment

Sterling Selling and Fulfillment Foundation provides integration with various material handling equipment (MHE). The automation enabled through the integration enables increased efficiency in various processes of a warehouse. For information about integrating Sterling Selling and Fulfillment Foundation with MHE, see “Overview of Material Handling Equipment Integration” on page 83.

Integration with Enterprise Resource Planning Systems

Sterling Selling and Fulfillment Foundation provides integration with the Enterprise Resource Planning (ERP) systems. An ERP system is a packaged business software system that allows a company to automate and integrate the majority of its business processes. For information about integrating Sterling Selling and Fulfillment Foundation with ERP Systems, see “Overview of Integration with ERP Components” on page 95.

Integration with Point of Sale Systems

Sterling Selling and Fulfillment Foundation provides integration with the point-of-sale systems used in stores for product check-outs and returns from customers. For information about integrating Sterling Selling and Fulfillment Foundation with point-of-sale systems, see “Integrating with Point of Sale Systems” on page 105.

Integration with JMS Systems

In order for some service nodes to communicate with external applications, external message queueing software must be configured. For information about configuring the third-party message queueing applications, see “Introduction to Integrating with JMS” on page 119.

Integration with Financial Systems

To use the data captured using the Sterling Selling and Fulfillment Foundation Inventory Cost Management feature with your financial system, you must load the Initial Inventory Cost Data and configure process-specific events.

For information about integrating Sterling Selling and Fulfillment Foundation with financial systems, see “Requirements for Financial System Integration” on page 131.

Rapid Deployment Features

Sterling Selling and Fulfillment Foundation Rapid Deployment Features can be utilized for the rapid deployment of Sterling Selling and Fulfillment Foundation. For information about Rapid Deployment Features, see “Rapid Deployment Feature Overview” on page 137.

Chapter 2. Integrating with the Distribution Center System

Introduction to Distribution Center System Integration

Note: For the Sterling Selling and Fulfillment Foundation Release 9.0, the integration described in this chapter has been deprecated.

The Distribution Center System (DCS) is a previously released product that supports warehouse activities such as the inventory of items and the distribution of packages. Typically, DCS operates in distribution centers fulfilling large numbers of orders for items required in quantities of a case or less. It supports both real-time radio frequency (RF) transactions and paper-based transactions.

Note: Sterling Selling and Fulfillment Foundation and Distribution Center System integration requires that the DCS interface format conforms to the field size and start positions at *each* of the integration points as detailed in the tables in this chapter. For information about configuring DCS, see the DCS 6.2 documentation. In addition, you must configure Sterling Selling and Fulfillment Foundation as described in this chapter.

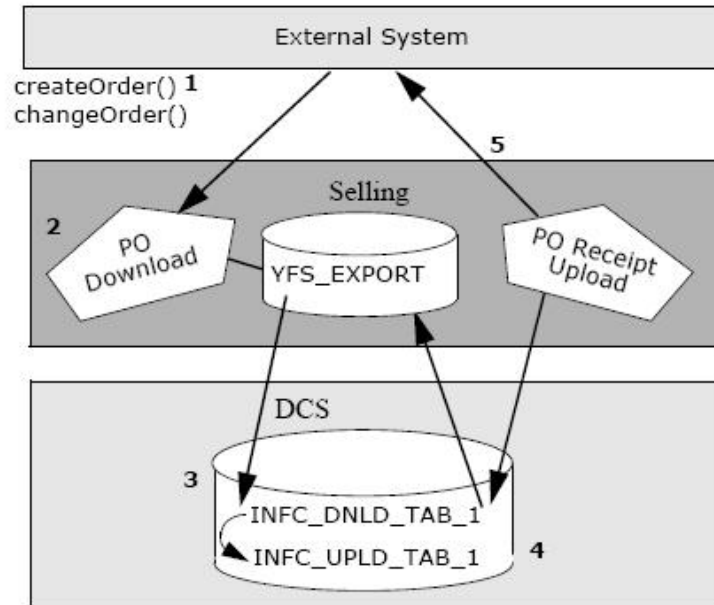
Note: Sterling Selling and Fulfillment Foundation is certified for DCS 6.2 Service Pack 3 Hot Fix 13 and above.

DCS Purchase Order Interface Function

When a Purchase Order is created on Sterling Selling and Fulfillment Foundation (either by importing Purchase Orders created by external order management systems or by using the Application Console), DCS integration enables you to publish that data to the DCS. The integration interface uses the Purchase Order Download and Upload time-triggered transactions. For more information about these transactions, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Purchase Order Workflow

The following figure illustrates the workflow for the Purchase Order Download and Purchase Order Upload time-triggered transactions that send Purchase Order data between an external system and the DCS using the Sterling Selling and Fulfillment Foundation.



1. An external Purchase Order system invokes Sterling Selling and Fulfillment Foundation `createOrder()` API to create a Purchase Order for a DCS receiving node. A Purchase Order is created and the order status becomes Created (1100). Any future modifications to the original Purchase Order by an external system are made by invoking the `changeOrder()` API.
2. The `ON_SUCCESS` event of the `createOrder()` or `changeOrder()` API invokes an action, which in turn invokes a service called `YantraWMSPoDownloadService`. This service publishes data into the `YFS_EXPORT` table with `YantraWMSPoDownloadService` as the flow name.
3. The Purchase Order Download time-triggered transaction takes the record from the `YFS_EXPORT` table and inserts it into the DCS interface `INFC_DNLD_TAB_1` table. Before downloading to the DCS, the transaction verifies that the ship node assigned to the Purchase Order line is a DCS ship node. If the ship node is not a DCS ship node, the transaction marks the record as processed and takes no further action.
4. The vendor sends an advance shipment notice (ASN) to the DCS for shipping the items on the Purchase Order. When items are received at the receiving node, the DCS uploads Purchase Order Receipt records to the DCS interface table.

The Purchase Order Receipt Upload time-triggered transaction picks up the Purchase Order Receipt records from the DCS upload interface table and calls the `receiveOrder()` API with the Receive Purchase Order transaction. The status of items received is changed to Received (3900).

If the Purchase Order is to be downloaded to Sterling Selling and Fulfillment Foundation from an external system, the `ON_SUCCESS` event of the Receive Purchase Order transaction can be configured to invoke an action to publish the Purchase Order Receipt data to the `YFS_EXPORT` table.

The data is then uploaded back to the external Purchase Order system.

For step-by-step procedures, see [Configuring Purchase Order Time-Triggered Transactions](#).

Understanding Purchase Order Transactions

When deciding how to implement DCS Purchase Order functionality, keep in mind the expected behaviors associated with the Purchase Order transactions in the contexts described in the following topics.

Supply Type Behavior

When the Purchase Order Status is Created (1100), the quantity in the YFS_INVENTORY_SUPPLY table is added to the PO_PLACED supply type.

When the Purchase Order Status is moved to Order Received (3900), the quantity in the YFS_INVENTORY_SUPPLY table moves from supply type PLANNED_PO to supply type ONHAND. This is the default behavior and can be reconfigured as needed.

Creating a Purchase Order

Sterling Selling and Fulfillment Foundation requires the Purchase Order number it passes to the DCS to be unique across all Enterprises. While Sterling Selling and Fulfillment Foundation permits the length of the Order number to be up to 40 characters, the DCS limits the length of both the Order and the Purchase Order number to a maximum of 13 characters. In addition, to comply with the DCS requirements, Purchase Order numbers may contain any combination of numbers and uppercase alphabetic characters; lowercase alphabetic characters are not permitted.

All Purchase Order lines must use consecutive prime line numbers, with all subline numbers as = 1. The PODTL Record Type does not take in subline numbers. For more information see “PODTL - Purchase Order Download Detail” on page 10.

When integrating with the DCS, all the advance shipment notifications (Purchase Order Receipt) created and uploaded to the DCS interface table are only for the Purchase Orders that were initially downloaded from Sterling Selling and Fulfillment Foundation.

When passing parameters to the DCS interface table, be sure that the length does not exceed that which is enabled by the DCS Purchase Order header and detail records.

Parameters are passed to the DCS when Sterling Selling and Fulfillment Foundation downloads Purchase Orders from an external system.

Note that the date for the Estimated Time of Arrival in the DCS is the Requested Delivery Date at the time of the Purchase Order creation on Sterling Selling and Fulfillment Foundation.

Modifying a Purchase Order

Only the following modifications to a Purchase Order are permitted:

- Changing the quantity
- Changing the requested delivery date
- Adding one or more lines

Splitting a Purchase Order

A Purchase Order cannot be split across receiving nodes, even for the same DCS. One Purchase Order is created for only one installation of the DCS and only one of its receiving nodes. All Purchase Order lines must have the same receiving node.

Canceling a Purchase Order or Line

While it is not possible to explicitly cancel a Purchase Order or Purchase Order line, if the quantity zero (0) is passed from Sterling Selling and Fulfillment Foundation, the Purchase Order modification time-triggered transaction interprets it as closing the order line on the DCS. For the DCS, the results of canceling a line is the same as closing a line. If the ordered quantity becomes zero, Sterling Selling and Fulfillment Foundation does not permit any further changes to the line.

If Sterling Selling and Fulfillment Foundation receives a Purchase Order receipt from the DCS on a line that has been cancelled by the external Purchase OrderPurchase Order system (due to interface timing issues), it raises an exception in Sterling Selling and Fulfillment Foundation.

Receiving Goods into Inventory

The warehouse receiving the goods is identified as the Receiving Ship Node on the Purchase Order.

The specific goods that a node receives must match the description of the line items on the original Purchase Order.

Receipt overage is controlled by the DCS by setting up an overage receipt percentage based on your receiving preferences for each line type downloaded.

Configure the Overage Receipt Percentage

About this task

To configure the overage receipt percentage in the Applications Manager:

Procedure

1. Navigate to Applications > Supply Collaboration > Document Specific > Purchase Order > Receipt > Receiving Preference.

2. On the Search Results panel choose 

The overage percentage is controlled in the DCS. The Sterling Selling and Fulfillment Foundation percentage is applied during receipt. This means that the receiving node for the DCS cannot receive quantity in excess of the overage percentage specified. Also, by the same logic, Sterling Selling and Fulfillment Foundation does not permit new order quantities to be modified to be below the quantity already received for that Purchase Order line.

3. Be sure to configure the received quantity so that Sterling Selling and Fulfillment Foundation and all the DCS work together.

For example, if received quantity is configured as ONHAND in Sterling Selling and Fulfillment Foundation, it should be configured as Allocatable in all the DCS installations.

Sterling Selling and Fulfillment Foundation and DCS Received Quantity Mapping

The following table shows *Sterling Selling and Fulfillment Foundation and DCS Received Quantity Mapping*:

Quantity Description	Sterling Selling and Fulfillment Foundation	DCS
Available items	ONHAND	Allocatable
Items kept in reserve	HELD	Non Allocatable

In addition, a node cannot receive goods against a cancelled or closed line.

Inventory is increased in the onhand supply when Sterling Selling and Fulfillment Foundation receives and processes the Purchase Order Receipt Upload transaction from the DCS, which must not be configured to upload separate inventory transactions for receipts.

For more information about configuring DCS Inventory Updates, see the DCS documentation.

Configuring Purchase Order Time-Triggered Transactions

About this task

Setting up a Purchase Order involves configuring and scheduling time-triggered transactions and configuring the pipeline that the Purchase Order should use. You also should check your Oracle database configuration.

To configure the Purchase Order time-triggered transactions:

Procedure

1. Check that Oracle database links are created for each DCS receiving node for which you want to create a Purchase Order. Sterling Selling and Fulfillment Foundation maintains the links and views to the DCS interface table for each receiving node in the DCS system.
2. Configure the Purchase Order Download and Purchase Order Receipt Upload time-triggered transactions. For detailed information about configuring these transactions, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Note: While the WMS Purchase Order Download time-triggered transaction does not require a ship node to be configured for downloading, you do need to configure agent criteria for each ship node from which a WMS Purchase Order Receipt Upload is to be processed.

3. Configure the pipeline using the directions in Configuring the Purchase Order Pipeline.
4. Schedule the time intervals for running the Purchase Order time-triggered transactions, as described in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

The Purchase Order Download transaction writes the POHDR and PODTL records into the DCS download interface table.

The Purchase Order Receipt Upload transaction reads the RCPHDR and RCPDTL records from the DCS upload interface table.

Configuring the Purchase Order Pipeline

About this task

The Purchase Order time-triggered transactions require a Purchase Order pipeline. If you need additional information about configuring pipelines, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

To configure the Purchase Order pipeline:

Procedure

1. From the Applications Manager menu, choose Business Process > Process Modeling.
2. Verify that the Purchase Order pipeline is configured with the following transactions:



3. At the bottom of the left pane, click the Services tab to open the Services tree.
4. Create a new service named YantraWMSPODownloadService that is invoked synchronously, does not provide real time response, and contains the following sequence of nodes:
 - a. Start node
 - b. Database node: specify the table name property as YFS_EXPORT
 - c. End node
5. Create an action. Click the Invoked Services tab and add the service YantraWMSPODownloadService you created.
6. Attach this action to the ON_SUCCESS events of the Create Order and Change Order transactions in the Purchase Order Execution repository. If necessary, add a condition to call this action only if the receiving node is the WMS Node.

DCS Purchase Order Interface

POHDR - Purchase Order Download Header

The following table lists the header information required by the Purchase Order Download time-triggered transaction.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	OrderLine.ReceivingNode in CreateOrder XML	5	1
record_type	'POHDR'	6	6
action_code	Always 'CH'	2	12
recv_order_type	'VN'	2	14
recv_order_no	Order.OrderNo in CreateOrder XML (Alphabetic characters must be upper-case)	13	16
recv_order_release_no	'1'	3	29
source	Order.SellerOrganizationCode in CreateOrder XML	10	32

PODTL - Purchase Order Download Detail

The following table lists the detail, or line information, required by the Purchase Order Download time-triggered transaction.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	OrderLine.ReceivingNode in CreateOrder XML	5	1
record_type	'PODTL'	6	6
action_code	'CL' Only for PO Line Close. (This happens when the line ordered quantity is reduced to zero.) 'CH' for all other modifications, such as changing the quantity (to nonzero), ETA, or adding lines.	2	12
recv_order_type	'VN'	2	14
recv_order_no	Order.OrderNo in CreateOrder XML (Alphabetic characters must be upper-case)	13	16
recv_order_release_no	'1'	3	29
recv_order_line_no	OrderLine.PrimeLineNo in CreateOrder XML	5	32
item_id	OrderLine.Item.ItemID in CreateOrder XML	24	37
product_class	OrderLine.Item.ProductClass in CreateOrder XML	6	61
pack_type	Always blank	4	67
order_qty	OrderLine.OrderedQty in CreateOrder XML	9	71
pre_production	Always blank	1	80
x_doc_recv_order	Always blank	1	81
eta_date	OrderLine.ReqShipDate in CreateOrder XML	8	82
unit_price	OrderLine.LinePriceInfo.UnitPrice in CreateOrder XML	11	90
country_of_origin	OrderLine.Item.CountryOfOrigin in CreateOrder XML	5	101
reference_1	Always blank	20	106
reference_2	Always blank	20	126
reference_3	Always blank	20	146

Sample Receive Order Output XML

The following code example shows a sample of the XML published by the ON_SUCCESS event of the Receive Order transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<Receipt EnterpriseCode="E1" OrderNo="BB_11" ReceiptNo="AMAR88891">
  <ReceiptLines>
    <ReceiptLine PrimeLineNo="2" Quantity="1.0" ReceiptHeaderKey=""
      SubLineNo="1"/>
  </ReceiptLines>
</Receipt>
```

RCPHDR - Purchase Order Receipt Header

The following table lists the header information required by the Purchase Order Receipt time-triggered transaction.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	OrderLine.ReceivingNode in CreateOrder XML	5	1
record_type	'RCPHDR'	6	6
action_code	Always 'AD'	2	12
asn_no	Advance Shipment Notice number	20	14
asn_type	Advance Shipment Notice type	2	34
reference_1	Reference number	30	191

RCPDTL - Purchase Order Receipt Detail

The following table lists the detail, or line information, required by the Purchase Order Receipt time-triggered transaction.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	OrderLine.ReceivingNode in CreateOrder XML	5	1
record_type	'RCPDTL'	6	6
action_code	Always 'AD'	2	12
asn_no	Advance Shipment Notice number	20	14
asn_type	Advance Shipment Notice type	2	34
recv_order_no	Order.OrderNo in CreateOrder XML (Alphabetic characters must be upper-case)	13	66
recv_order_line_no	OrderLine.PrimeLineNo in CreateOrder XML	5	82
received_qty	Quantity received in ASN against order line number	7	119

Receive Order Input XML Mapping

The receiveOrder() API input XML maps to DCS tables at the order header level and at the order line level.

Order Header Records

The receiveOrder() API input XML and the DCS Order Header map as shown in the following table:

Table 1. Order Header Records

Sterling Selling and Fulfillment Foundation XML Parameter	DCS Parameter
orderheaderkey	Always blank
orderreleasekey	Always blank
receiptheaderkey	Always blank
receiptno	HEADER.ANS_NO
releasenno	Always blank

Shipment Records

The receiveOrder() API input XML and the DCS Order map as shown in the following table:

Table 2. Shipment Records

Sterling Selling and Fulfillment Foundation XML Parameter	DCS Parameter
enterprisecode	EnterpriseCode
orderno	DETAIL.RECV_ORDER_NO

Order Line Records

The receiveOrder() API input XML and the DCS Order Line map as shown in the following table:

Table 3. Order Line Records

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter
BreakIntoComponents	Always blank
DispositionCode	Always blank
InspectedBy	Always blank
InspectionComments	Always blank
InspectionDate	Always blank
LotNumber	Always blank
OrderLineKey	Always blank
PrimeLineNo	DETAIL.RECV_ORDER_LINE_NO
SubLineNo	1
Quantity	DETAIL.RECEIVED_QTY
ReceiptLineNo	Always blank
SerialNo	Always blank
ShipByDate	Always blank
<KitLines>	Not Used

DCS Shipment Interface

The DCS integrates with the IBM Sterling Distributed Order Management interface of Sterling Selling and Fulfillment Foundation. This integration enables shipment-related information to be passed between applications.

Understanding Order Transactions

Before implementing the upload and download functionality, you should understand the following default behaviors:

- Modifications to an Order or Order Release in Sterling Selling and Fulfillment Foundation after download to DCS are not transmitted to DCS.

- Inventory is reduced from the onhand supply when Sterling Selling and Fulfillment Foundation receives and processes the shipment confirmation transaction from DCS. DCS must not be configured to upload separate inventory transactions for shipments.
- The SCAC and Service Code used by the Sterling Selling and Fulfillment Foundation input XML corresponds to the SCAC field in the DCS interface. Map each carrier defined in DCS to those in Sterling Selling and Fulfillment Foundation by creating an identical configuration in the Applications Manager > Application Platform > Participant Modeling. For example, if DCS uses *UPSG* as the SCAC Code for United Parcel Ground Service, in Sterling Selling and Fulfillment Foundation for the participant called *UPS*, set the SCAC and Service Code as *UPSG*, and specify the *Service* as *Ground*.
- DCS should disable cancellation from transaction 02012 (Order Release list). Sterling Selling and Fulfillment Foundation only recognizes cancellations with return ownership = Y when done from DCS transaction 02013 (load/shipper list).
- The Order No for Shipment Advice can be a maximum length of 13 bytes and must be upper-case characters and numbers or just numbers (lowercase characters are not allowed).

Configuring DCS Shipment Time-Triggered Transactions

About this task

Setting up a sales order involves configuring and scheduling the Send Release and WMS Shipment Confirmation time-triggered transactions and configuring the pipeline a sales order should use. You also should check your Oracle database configuration.

To configure the Send Release and WMS Shipment Confirmation time-triggered transactions:

Procedure

1. Check your Oracle database to ensure that links are created for each DCS ship node for which you create a Release. Sterling Selling and Fulfillment Foundation maintains links and views to the DCS interface tables for each node.
2. Configure the Send Release and Ship Confirm time-triggered transactions:
 - If you want to configure the Send Release transaction, from the Applications Manager Applications menu, choose Application Platform > Process Modelling > Order > Sales Order > Order Fulfillment > Transaction Repository Send Release.
 - If you want to configure the Ship Confirm transaction, from the Applications Manager Applications menu, choose Application Platform > Process Modelling > General > General > Transaction Repository > Ship Confirm.

Note: While the Send Release time-triggered transaction does not require a ship node to be configured for downloading, you do need to configure agent criteria for each ship node from which a WMS shipment confirmation is to be processed.

3. Configure the Sales Order Fulfillment Pipeline to download ship advice to DCS and receive shipment confirmation from DCS.

The repository has a default pipeline configured to download shipment advice to DCS and receive shipment confirmation. When modifying the pipeline, first

copy the default pipeline and then modify that copy to suit your needs. For more information about configuring a pipeline, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

While configuring the pipeline, keep in mind the following characteristics of the DCS shipment-related integration:

- Order Releases to be downloaded to DCS are staged with the status Awaiting WMS Interface (3200.02). The Send Release transaction in the pipeline is configured to pick up these Order Releases and download them to DCS.
 - After the download completes, the Order Release status moves to Sent to Node (3300).
 - The Shipment Confirmation transaction uploads the shipment from the DCS interface table and moves the status of the Order to Shipped (3700).
4. Schedule the time intervals for running the Send Release and WMS Shipment Confirmation time-triggered transactions from DCS, as described in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

DCS Order Release Interface

When Order Releases are going to the DCS interface, the Send Release transaction dispatches the Order Release information to DCS in open interface format if the ship node's interface is set to DCS in Sterling Selling and Fulfillment Foundation.

Setting a Ship Node as a WMS Ship Node

About this task

To set the ship node as a WMS ship node:

Procedure

1. From the Applications Manager Applications menu, choose Application Platform > Participant Modeling > Organization Details > Roles & Participation Tab > Node Attributes/Primary Info Tab (on the right) > Execution In Node Using.
2. Choose the Sterling Warehouse Management System 6.2.

Send Release Transaction Supported Record Types

Sterling Selling and Fulfillment Foundation supports the following record types, which are output by the Send Release time-triggered transaction:

- ORDHDR – Order Release Order Header
- ORDDTL – Order Release Order Detail
- ORDADR – Order Release Order Address
- ORDINS – Order Release Order Instruction
- ORDBOM – Order Release Order Bill of Materials
- ORDNAM – Order Release Order Name

These record types are written by the Send Release transaction into the DCS download interface table.

Only the action code Add (AD) is supported by Sterling Selling and Fulfillment Foundation.

ORDHDR – Order Release Order Header

The following table lists the Order Release header information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'ORDHDR'	6	6
action_code	Always 'AD'	2	12
order_no	Order Number (Alphabetic characters must be upper-case)	13	14
order_rel_no	Order Release Number	3	27
ship_to_cust_id	PersonInfoShipTo PersonID in CreateOrder XML	35	30
bill_cust_id	PersonInfoBillTo PersonID in CreateOrder XML	35	65
forward_to_cust_id	PersonInfoMarkFor PersonID in CreateOrder XML	35	100
pack_hold_flag	Always 'N'	1	135
order_type	OrderType in CreateOrder XML	1	136
order_cancel_date	ReqCancelDate in Order Release being downloaded	8	147
order_due_date	ReqDeliveryDate in CreateOrder XML	8	155
terms_code	TermsCode in CreateOrder XML	8	163
carrier_code	SCAC	4	173
priority_code	PriorityCode in CreateOrder XML	2	177
consol_rule	Always blank	2	179
cartonization_rule	Always blank	2	181
cust_order	CustomerPONo in CreateOrder XML	25	183
pack_list_type	PackList Type in ShipAdvice XML	2	208
spc_ticket_req	PersonalizeCode in CreateOrder XML	2	210
asn_flag	NotifyAfterShipmentFlag in CreateOrder XML	1	212
delivery_date	ReqDeliveryDate in Order Release being downloaded	8	216
orig_ship_date	ReqShipDate in Order Release being downloaded	8	224
samples_flag	Always blank	1	234
ship_to_customer_name	PersonInfoShipTo FirstName and LastName in CreateOrder XML	35	235
ship_to_addr1	PersonInfoShipTo AddressLine1 in CreateOrder XML	35	270
ship_to_addr2	PersonInfoShipTo AddressLine2 in CreateOrder XML	35	305
ship_to_addr3	PersonInfoShipTo AddressLine3 in CreateOrder XML	35	340
ship_to_addr4	PersonInfoShipTo AddressLine4 in CreateOrder XML	35	375
ship_to_city	PersonInfoShipTo City in CreateOrder XML	30	410

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
ship_to_state	PersonInfoShipTo State in CreateOrder XML	2	440
ship_to_zip_code	PersonInfoShipTo Zip Code in CreateOrder XML	9	442
ship_to_country_code	PersonInfoShipTo Country in CreateOrder XML	5	451
cross_dock_flag	Always blank	2	456
split_flag	ShipCompleteFlag in CreateOrder XML	1	488
consol_flag	Always blank	1	489
shippable_order	Always 'Y'	1	490
delivery_code	DeliveryCode in CreateOrder XML	1	517
back_order_authorized_ind	Always '01'	2	526
cal_check_req_ind	Always 'N'	1	541
inbound_flag	Always 'N'	1	550
order_create_date	OrderDate in CreateOrder XML	8	564
carrier_service	Carrier Service Code in CreateOrder XML	10	572
cust_carrier_charge_account_no	Carrier Account Number in CreateOrder XML	35	582
enterprise_code	Enterprise Code	24	639

ORDDTL – Order Release Order Detail

The following table lists the Order Release detail information mapped to DCS.

Table 4. Order Release Detail Information Mapped to DCS

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'ORDDTL'	6	6
action_code	Always 'AD'	2	12
order_no	Order Number (Alphabetic characters must be upper-case)	13	14
order_rel_no	Order Release Number	3	27
order_prime_line	Order Prime Line Number	5	30
order_sub_line	Order Sub Line Number	5	35
mark_for	PersonInfoMarkFor PersonID in CreateOrder XML	35	40
item_id	ItemID in CreateOrder XML	24	75
product_class	ProductClass in CreateOrder XML	6	99
quality_status	Always blank	2	105
department_code	DepartmentCode in CreateOrder XML	6	107
hazard_flag	Always 'N'	1	119
qty_ordered	OrderedQty in CreateOrder XML	9	120
shippable_qty	Total Quantity to be shipped	9	129

Table 4. Order Release Detail Information Mapped to DCS (continued)

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
nonshippable_qty	Always '0'	9	138
pack_type	Always 'EACH'	4	147
ship_together_code	ShipTogetherNo in CreateOrder XML	5	151
line_price	Unit Price from LinePriceInfo in CreateOrder XML	11	156
spl_processing_code1	Always blank	4	167
orig_req_ship_date	ReqShipDate in CreateOrder XML	8	249
act_req_ship_date	ReqShipDate in CreateOrder XML	8	257
customer_po_no	CustomerPONo in CreateOrder XML	25	269
ship_sure_model_ind	Always 'Y'	1	294
order_line_point	Always blank	5	295
line_type	Always blank	4	300
carrier_code	Always blank	4	304
samples_flag	Always 'N'	1	308
customer_po_line_no	CustomerLinePONo in CreateOrder XML	13	335
customer_sku	CustomerItem in CreateOrder XML	40	386
kit_code	KitCode in CreateOrder XML	2	466

ORDADR – Order Release Order Address

The following table lists the Order Release address information mapped to DCS.

Table 5. Order Release Address Information Mapped to DCS

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'ORDADR'	6	6
action_code	Always 'AD'	2	12
order_no	Order Number (Alphabetic characters must be upper-case)	13	14
order_rel_no	Order Release Number	3	27
address_type	'FT' or 'BT'	2	30
customer_name	FirstName and LastName in CreateOrder XML	35	32
addr1	AddressLine1 in CreateOrder XML	35	67
addr2	AddressLine2 in CreateOrder XML	35	102
addr3	AddressLine3 in CreateOrder XML	35	137
addr4	AddressLine4 in CreateOrder XML	35	172
city	City in CreateOrder XML	30	207
state	State in CreateOrder XML	2	237
zip_code	Zip Code in CreateOrder XML	9	239

Table 5. Order Release Address Information Mapped to DCS (continued)

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
country_code	Country in CreateOrder XML	5	248
wms_buffer	Always blank	30	253

ORDINS – Order Release Order Instruction

The following lists the Order Release instruction information mapped to DCS.

Table 6. Order Release Instruction Information Mapped to DCS

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'ORDINS'	6	6
action_code	Always 'AD'	2	12
order_no	Order Number (Alphabetic characters must be upper-case)	13	14
order_rel_no	Order Release Number	3	27
order_prime_line	Order Prime Line Number	5	30
order_sub_line	Order Sub Line Number	5	35
instruction_type	InstructionType in CreateOrder XML	3	40
seq_no	Sequence Number of instructions	3	43
usage_type	Instruction usage	2	46
instructions_text	InstructionText in CreateOrder XML	80	48
wms_buffer	Always blank	30	128

ORDBOM – Order Release Order Bill of Materials

The following table lists the Order Release Bill of Materials information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'ORDBOM'	6	6
action_code	Always 'AD'	2	12
order_no	Order Number (Alphabetic characters must be upper-case)	13	14
order_rel_no	Order Release Number	3	27
order_prime_line	Order Prime Line Number	5	30
order_sub_line	Order Sub Line Number	5	35
item_id	ItemID in CreateOrder XML	24	40
product_class	ProductClass in CreateOrder XML	6	64
quality_status	Always blank	2	70

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
pack_type	Always 'EACH'	4	72
bom_qty	KitQty in CreateOrder XML	9	76
pick_slip_number	Always blank	13	85
picking_line_detail_id	Always blank	13	98
scrap_factor	Always '0000000'	7	111
reference_field1	Always blank	40	118
reference_field2	Always blank	40	158
reference_field3	Always blank	40	198
reference_field4	Always blank	40	238
reference_field5	Always blank	40	278
wms_buffer	Always blank	30	318

ORDNAM – Order Release Order Name

This interface format is used to send orders having the following information:

- COD - This record is sent for orders having PaymentType as COD.
- Customer Phone Number - This record is sent only if the ShipTo Customer Day Phone Number is not blank.
- Importer information - This record is sent for international shipments only. This information is not sent if country code in any address (ship node or ship-to address) is blank.
- YFS accepts Import License ID and Import License Expiration Date at Order line level, whereas DCS accepts it at Order header level.
- Exporter Information - This record is sent for international shipments only.

The ship node address country code and ship-to address country code should not be blank.

The following table lists the Order Release name information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'ORDNAM'	6	6
action_code	Always 'AD'	2	12
order_no	Order Number (Alphabetic characters must be upper-case)	13	14
order_rel_no	Order Release Number	3	27
name	For COD- '100' For Customer Phone Number - '300' For Importer Information - '400' ¹ For Exporter Information - '400'	3	30

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
value	For COD - '103' For Customer Phone Number - '301' For Importer Information - '402' ¹ For Exporter Information - '401'	3	33
reference_field1	For COD - 'COD' For Customer Phone Number - PersonInfoShipTo DayPhone in CreateOrder XML For Importer information - TaxPayerId in CreateOrder XML ¹ For Exporter Information - ExportTaxPayerId of the ShipNode	40	36
reference_field2	For COD - Always blank For Customer Phone Number - Always blank For Importer Information - ImportLicenseNo in CreateOrder XML ¹ For Exporter Information - ExportLicenseNo of the ShipNode	40	76
reference_field3	For COD - Always blank For Customer Phone Number - Always blank For Importer information - ImportLicenseExpDate in CreateOrder XML ¹ For Exporter Information - ExportLicenseExpDate of the ShipNode	40	116
reference_field4	Always blank	40	156
reference_field5	Always blank	40	196
wms_buffer	Always blank	30	236

Tracking License Information

When Sterling Selling and Fulfillment Foundation sends Order Release information to DCS, it sends only the Import License ID and Import License Expiration Date from the first order line and ignores information from the other lines. As a result, if you need to track all license information, group items by license type in separate orders.

For example, put all materials that require the same type of license for hazardous material on one order and items that require the same type of license for nonhazardous chemicals on another.

DCS Shipment Confirmation

Sterling Selling and Fulfillment Foundation picks up the shipment confirmations posted by DCS in the open interface tables. The WMS Shipment Confirmation time-triggered transaction performs shipment confirmation.

Only action codes Cancel (CA) and Ship (SH) are picked up by Sterling Selling and Fulfillment Foundation.

Not all records and attributes are supported by Sterling Selling and Fulfillment Foundation. The WMS Shipment Confirmation transaction reads only the PCKHDR, CARHDR, PCKINF, CNCDTL, and SRLDTL record types from the DCS upload interface table.

PCKHDR – Shipment Confirmation Picketicket Header

The following table lists the shipment confirmation pickticket header information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'PCKHDR'	6	6
action_code	Always 'CA' or 'SH'	2	12
pickticket_no	PickTicketNo in confirmShipment XML	20	14
ship_type	Ship Mode in confirmShipment XML	4	80
actual_ship_date	ShipDate in confirmShipment XML	8	92
carrier_code	SCAC and Service Code in confirmShipment XML	4	107
trailer_no	TrailerNo in confirmShipment XML	20	111
freight_charges	FreightCharge in confirmShipment XML	13	131
manifest_no	ManifestNo in confirmShipment XML	20	144
bol_no	BOL Number	20	164
pro_no	ProNo in confirmShipment XML	20	184
master_bol_no	Parent Shipment Key	20	204
total_weight	TotalWeight in confirmShipment XML	13	224
seal_no	Seal Number	20	250
total_volume	TotalVolume in confirmShipment XML	7	296
it_number	IT number	20	303
it_date	IT Date	8	323
from_appointment_date	From appointment date	8	331
to_appointment_date	To appointment date	8	339
appointment_number	Appointment number	40	363
ship_to_addr1	ToAddress AddressLine1 in confirmShipment XML	35	483
ship_to_addr2	ToAddress AddressLine2 in confirmShipment XML	35	518
ship_to_addr3	ToAddress AddressLine3 in confirmShipment XML	35	553

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
ship_to_addr4	ToAddress AddressLine4 in confirmShipment XML	35	588
ship_to_city	ToAddress City in confirmShipment XML	30	623
ship_to_state	ToAddress State in confirmShipment XML	2	653
ship_to_zip	ToAddress Zip Code in confirmShipment XML	9	655
ship_to_country	ToAddress Country in confirmShipment XML	5	664

CARHDR – Shipment Confirmation Carton Header

The following table lists the carton header information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'CARHDR'	'6	6
action_code	'CA' or 'SH'	'2	12
picketticket_no	Not used	20	14
carton_no	Container number	11	34
weight	Used for containers other than pallets. Used as Container Gross Weight and Container Net Weight.	13	59
tracking_number	Tracking number	20	72
ucc128_code	If the third character is not '1', this is used as Container SCM.	30	92
manifest_no	Manifest number	10	122
pallet_scm	If the third character of ucc128_code is '1', this is used as Container SCM.	30	132
package_type	Container type	2	162
pallet_length	Used if the container is pallet. Specifies the pallet length.	9	164
pallet_width	Used if the container is pallet. Specifies the pallet length.	9	173
pallet_height	Used if the container is pallet. Specifies the pallet height.	9	182
pallet_gross_weight	Used if the container is pallet. Specifies the pallet gross weight.	9	191
pallet_net_weight	Used if the container is pallet. Specifies the pallet net weight.	9	200
carton_length	Used for containers other than pallet. Specifies the container length.	9	209
carton_width	Used for containers other than pallet. Specifies the container width.	9	218

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
carton_height	Used for containers other than pallet. Specifies the container height.	9	227
freight_charge	Freight Charge	7	238

PCKINF – Shipment Confirmation Pick Information

The following table lists the shipment confirmation pick information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'PCKINF'	6	6
action_code	Always 'CA' or 'SH'	2	12
pickticket_no	PickTicketNo in confirmShipment XML	20	14
carton_no	Container No	11	34
item_id	Item ID	24	45
product_class	Product Class	2	69
picked_qty	Shipped Qty	9	80
order_no	Order No	13	89
order_release_no	Release Number	3	102
order_line_no	Prime Line No	4	105
sub_line_no	Sub Line No	5	109

CNCDDL – Shipment Confirmation Cancel Detail

The following table lists the cancel detail information mapped to DCS.

The CNCDDL record is created only when Orders or Shipments are cancelled or backordered from the DCS Load/Shipper screen (02013), not the Order Release List screen (02012).

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'CNCDDL'	6	6
action_code	Always 'CA' or 'SH'	2	12
order_no	Order No	13	34
order_release_no	Order Release No	3	47
order_line_no	Prime Line No	5	50
sub_line_no	Sub Line No	5	55
item_id	Item ID	24	60
product_class	Product Class	2	84
cancel_quantity	BackOrder Qty	9	92

SRLDTL - Pick Ticket Serial Record

Sterling Selling and Fulfillment Foundation can accept serial numbers when an item has been configured in DCS as Serialized and the Sterling Selling and Fulfillment Foundation WMS Ship Confirmation agent is used. When an item is configured as Serialized in DCS and is shipped from DCS, DCS publishes SRLDTL records into the interface tables.

The WMS Ship Confirm Upload agent reads the interface records published by DCS and forms an input XML for the `confirmShipment()` API.

The SRLDTL records published by DCS are across order lines. These records do not contain line information. Sterling Selling and Fulfillment Foundation retrieves the serial records corresponding to each shipment line by matching the following attributes from the SRLDTL record with the shipment line, and making a subset of serial records for each shipment line:

- Item ID of SRLDTL with item id of Shipment line,
- Product Class of SRLDTL with product class of Shipment line,
- Pallet SCM of SRLDTL with pallet SCM on the container for the shipment line.
- Carton SCM: Based on setup in DCS, this field can have either carton SCM or container number. If the attribute length is 20, it is mapped to the Carton SCM of the shipment line. Otherwise, it is mapped to the Container Number of the shipment line.

Once a subset of the SRLDTL records is formed, Sterling Selling and Fulfillment Foundation adds a `ShipmentLine` element for each SRLDTL record in the XML and reduces the quantity from the already existing `ShipmentLine` element.

For example, Not Used if a shipment line has five units and there are five SRLDTL records for each unit, Sterling Selling and Fulfillment Foundation adds five `ShipmentLine` elements to the input XML and reduces the quantity of the original element to zero (0).

Note that the `YFS_Container_Details` table should have a serial number for each unit shipped.

Pickticket Serial Record Information Mapped to DCS

The following table lists the pickticket serial record information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode (Not used)	5	1
record_type	SRLDTL	6	6
action_code	Not used	2	12
pickticket_no	Not used	20	14
item_id	Shipment line's item ID	24	34
product_class	Shipment line's product class	2	58
item_pseudo_no	Not used	12	60
item_serial_no	Serial number (Passed to the API)	20	72
component_item_id	Not used	24	92
component_product_class	Not used	2	116

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
component_pseudo_no	Not used	12	118
component_serial_no	Not used	20	130
quantity	Quantity	9	150
country_of_origin	Not used	5	159
customer_po_number	Not used	25	164
pallet_scm	CARHDR's pallet SCM.	20	189
carton_scm	If the attribute length is 20, it is mapped to the Carton SCM of the shipment line. Otherwise, it is mapped to the Container Number of the shipment line.	20	209
upc_code	Not used	12	229
upc_case_code_scanned	Not used	14	241
upc_case_code_number_of_boxes	Not used	7	255

DCS Inventory Interface

The DCS inventory interface can download inventory changes due to Returns in Sterling Selling and Fulfillment Foundation to DCS. It can also read the uploads of inventory changes from DCS to Sterling Selling and Fulfillment Foundation.

DCS Inventory Upload

The Sterling Selling and Fulfillment Foundation inventory upload picks up inventory change information from DCS and uploads the information to Sterling Selling and Fulfillment Foundation. The WMS Inventory Upload time-triggered transaction, scheduled through `yfs.wms.inventory`, performs inventory change uploading which is read by the WMS Inventory Upload transaction.

DCS passes only one record type, TRNDTL, for an item and product class combination.

TRNDTL – Inventory Change Upload Record

The following table lists the inventory change upload information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	ShipNode	5	1
record_type	Always 'TRNDTL'	6	6
action_code	Always 'AD'	2	12
tran_code	ReferenceField4	5	31
tran_reason_code	ReasonCode	4	41
item_id	Item ID	24	45
product_class	Product Class	2	69
pack_type	UOM	4	71
unavailable_quantity	HeldQty	8	77

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
available_quantity	OnHandQty	8	85
held_quantity	HeldQty	8	93
order_release_no	ReferenceField2	3	203
order_line_no	ReferenceField3	5	206
to_location	ReferenceField5	12	284
reference_field_1	ReferenceField1	30	296
reference_field_2	ReasonText	30	326

Note: Reference Fields 1-5 map to the reference field in the YFS_Inventory_Audit table.

Note: For Work Orders, DCS sets the value of the Reference_Field4 Sterling Selling and Fulfillment Foundation Parameter to KITD for use by inventory costing.

For more information about the Inventory Upload transaction, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

DCS Inventory Download

When a Return is recorded in the Sterling Selling and Fulfillment Foundation, inventory adjustments may take place depending on the configuration of Status and Supply Type. When inventory adjustments take place on ship nodes specified as InterfaceType DCS, the inventory changes are published to the WMS interface tables, if a service is configured to do so.

Note: Do not configure multiple Supply Types to be downloaded to DCS. Doing so downloads duplicate records to the INFC_DNLD_TAB_1 interface table.

Configuring an Inventory Download Service About this task

To configure an inventory download service:

Procedure

1. From the Applications Manager menu, choose Business Process > Process Modeling. Open the General tab and choose the Details of the General process type.
2. Create a new service that is invoked synchronously, does not provide real time response, and contains the following sequence of nodes:
 - a. Start node.
 - b. API node. Choose Extended API node and configure it as follows:
 - Specify any name for API name.
 - Specify Class Name as `com.yantra.inv.business.inventory.YFSInventoryDownload`
 - Specify Method Name as `downloadInventory`
 - c. End node.
3. Create an action. Choose the Invoked Services tab and add the service you created.

4. Enable the INVENTORY_CHANGE event raised by the INVENTORY_CHANGE transaction.
5. Attach the action created in 3 on page 27 to the INVENTORY_CHANGE event of the INVENTORY_CHANGE transaction.
6. If necessary, add a Condition node to call the action only if AdjustmentType is RETURN. The AdjustmentType is RETURN when inventory adjustments take place due to Returns.

Note: Even if a service is configured unconditionally, the ship node must be specified as InterfaceType DCS and AdjustmentType is RETURN in order for the data to be written to the interface tables.

Input XML Passed to the Inventory Download Service: The following input XML is passed to the Inventory Download service by the event:

```
<?xml version="1.0" encoding="UTF-8"?>
<YantraXML>    <XML AccountNo="" AdjustmentType=" "
                CostCurrency="" EnterpriseCode=" " ItemID=" "
                ItemKey="" Organization=" "
                ProductClass="" Quantity="" ReasonCode="" ReasonText=""
                Reference_1=""
                Reference_2="" Reference_3="" Reference_4=""
                Reference_5="" ShipByDate="" ShipNode=""
                SupplyReference=" " SupplyReferenceType=""
                SupplyType=" " UnitCost="" UnitOfMeasure=" " /> </YantraXML>
```

The downloadInventory() method publishes inventory to WMS only if the 'AdjustmentType' in the XML is 'RETURN' and the ship node's interface type is 'WMS_YANTRA'. This method converts the XML into a WMS format string. A record is inserted into the 'Infc_Dnld_Tab_1' table in the WMS database with interface type as 'INVD'.

Adding Location and Reference Fields

The default XML (published by the event) does not contain location. Either Sterling Warehouse Management System can be configured to have a default location or this XML can be modified (to add the 'WarehouseLocation' attribute) in the service before passing it to this method. If the XML contains the 'WarehouseLocation' attribute, it is passed to Sterling Warehouse Management System as the location. Similarly, the 'WMSReferenceField1' and 'WMSReferenceField2' attributes can be added to the XML for Sterling Warehouse Management System fields 'ReferenceField1' and 'ReferenceField2'.

INVCHG - Inventory Change Download Record

The following table lists the inventory change download information mapped to DCS.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
whse	/YantraXML/XML/ShipNode	5	1
record_type	INVCHG	6	6
action_code	AD	2	12
tran_date	Transaction date in 'CCYYMMDD' format	8	14
tran_time	Transaction time in 'HHMMSS' format	6	22
tran_seq_no	001	3	28

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
tran_code	WIMT	5	31
tran_type	Always blank	5	36
tran_reason_code	Always blank	4	41
item_id	/YantraXML/XML/ItemID	40	45
product_class	/YantraXML/XML/ProductClass	6	85
pack_type	EACH	4	91
quality_status	Always blank	2	95
unavailable_quantity	0	7	97
available_quantity	/YantraXML/XML/Quantity	7	104
held_quantity	0	7	111
location	/YantraXML/XML/WarehouseLocation	20	118
user_id	User ID from the context	8	138
reference_field_1	Data maps to /YantraXML/XML/WMSReferenceField1. No data is passed, it maps to /YantraXML/XML/SupplyReference. Note: OrderNo is passed in /YantraXML/XML/SupplyReference by the event.	30	146
reference_field_2	Data maps to /YantraXML/XML/WMSReferenceField2. No data is passed, it maps to /YantraXML/XML/SupplyType.	30	176
wms_buffer	Always blank	30	206

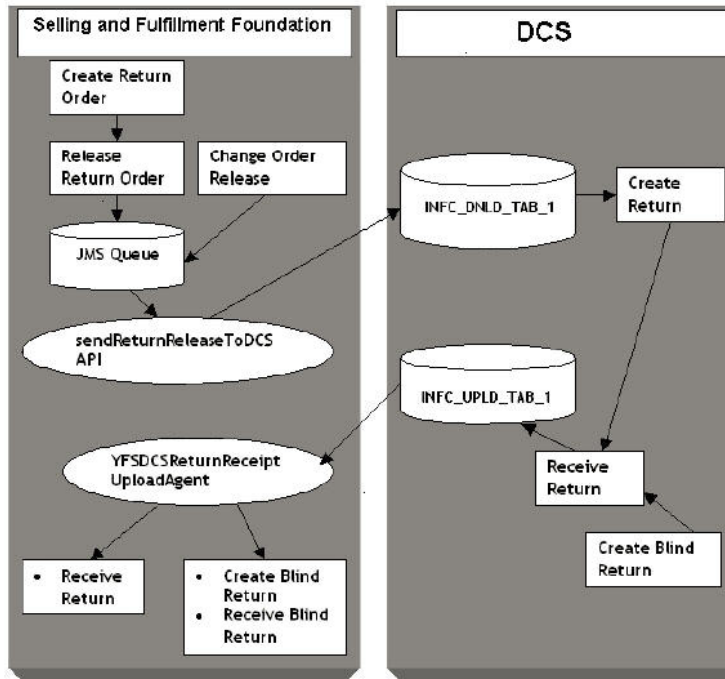
DCS Returns Interface

The integration of Sterling Selling and Fulfillment Foundation with DCS enables information related to Return Order release to pass between the two applications.

The integration provides an API (sendReturnReleaseToDCS) to send the Return Order release to DCS, and a time triggered transaction (DCS Return Receipt Upload Agent) to get the return release receipt information from DCS. Additionally, this integration supports receipts against blind returns that were created on DCS.

Return Order Integration Workflow

The following figure illustrates the workflow for the Return Order integration.



1. An external Return Order system invokes the createOrder() API on Sterling Selling and Fulfillment Foundation to create a Return Order for a DCS receiving node. A Return Order is created and the order status becomes Created (1100).
2. When the Return Order is released, the ON_RELEASE_CREATION_OR_CHANGE event of the releaseOrder API can be configured to invoke the service YantraSendReturnReleaseToDCSService, which inserts a message containing the return release key into the JMS Queue. For more information about configuring Return Order integration with DCS, see "Configuring Return Order Integration with DCS".

The return release is modified by invoking the changeRelease API. After the return release modification, if the ON_SUCCESS event is configured to invoke YantraSendReturnReleaseToDCSService, a message containing the release key is inserted into the JMS queue.

3. The sendReturnReleaseToDCS API picks up the return release key from the JMS Queue, fetches the release details, and inserts a message containing the release details into the DCS interface table INFC_DNLD_TAB_1.
4. An agent on DCS picks up the return release data from INFC_DNLD_TAB_1 and creates a return on DCS.
5. Alternatively, a blind return can be directly created on DCS using the DCS user interface.
6. Once the return is received, DCS agents insert the receipt details into the interface table INFC_UPLD_TAB_1.
7. The DCS Return Receipt Upload Agent picks up the receipt details from the interface table INFC_UPLD_TAB_1 and calls the receiveOrder API to mark the Return Order as received.

For blind returns, before calling the receiveOrder API, the DCS Return Receipt Upload Agent first calls the createOrder API to create a Return Order, or the

changeOrder API to change the order that already exists for this blind return on Sterling Selling and Fulfillment Foundation.

Determining the Enterprise Code for Blind Return During Upload

For blind RMA the system determines the enterprise code as follows:

1. If the value of RARHDR.REFERENCE-1 is blank, the primary organization of the owner of the ship node is taken as the enterprise code.
2. If the value of RARHDR.REFERENCE-1 is not blank, the system checks the value of RARHDR.REFERENCE-1.
 - If the value of RARHDR.REFERENCE-1 is a valid organization with an Enterprise role, the system uses the value of RARHDR.REFERENCE-1 as the enterprise Code.
 - If the value of RARHDR.REFERENCE-1 is not a valid organization with an Enterprise role, the system throws an error.

Configuring Return Order Integration with DCS

Return Order Integration with DCS can have various configurations.

The setup for the Disposition Code should be identical in both Sterling Selling and Fulfillment Foundation and DCS.

Inventory updates during return receipt upload should be turned off. Inventory adjustments for return receipts should be done through the inventory adjustment interface with DCS. Whenever inventory is updated in DCS, inventory is updated in Sterling Selling and Fulfillment Foundation too through this interface.

Configuring Return Release Download to DCS

About this task

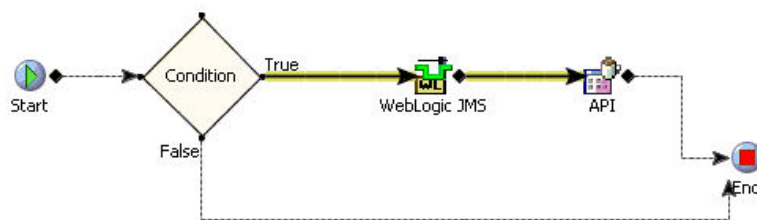
Configuring return release download to DCS involves creating a new JMS Queue, service, and action.

To configure return release download to DCS:

Procedure

1. Create a synchronous service such as YantraSendReturnReleaseToDCSService under Reverse Logistics Services.

This service puts the Return Order release key into a JMS queue, say RMADownloadQueue, if the ship node is a DCS node.



The "Condition" mentioned in the figure should be configured with ship node interface type = 'WMS_YANTRA'.

2. For the API component in the service,
 - Choose the General tab.

- Select the Sterling Selling and Fulfillment Foundation Standard API option button.
- From the API Name drop-down list, select sendReturnReleaseToDCS.

When the integration server configured in the JMS receiver runs, the sendReturnReleaseToDCS API picks up the order release key from the JMS queue and inserts the return release details in the DCS interface table.

3. Navigate to ReverseLogistics Repository > Actions and create an action, say SendReturnReleaseToDCS.

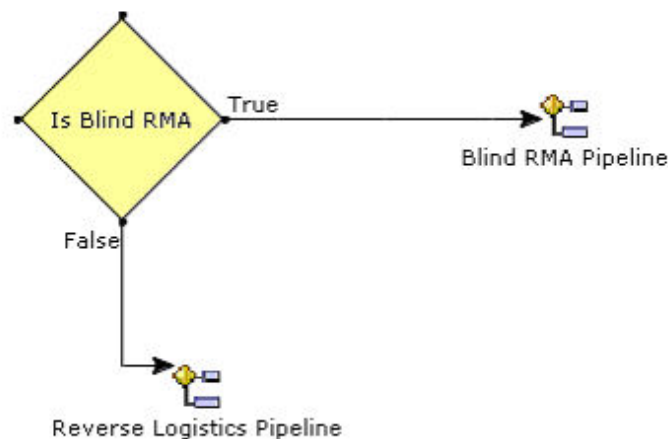
This action should invoke YantraSendReturnReleaseToDCSService.

4. Configure ON_RELEASE_CREATION_OR_CHANGE event of the SCHEDULE RETURN transaction and ON_SUCCESS event of the changeRelease API to invoke this action (in the case of Reverse Logistics, the SCHEDULE RETURN transaction also does the release).

Configuration for Receiving Blind RMA

Return Receipts for Blind RMAs created at the warehouse and the receipt details are uploaded as regular return receipts. The receipt upload agent creates the Return Order with a '03' order type in Sterling Selling and Fulfillment Foundation.

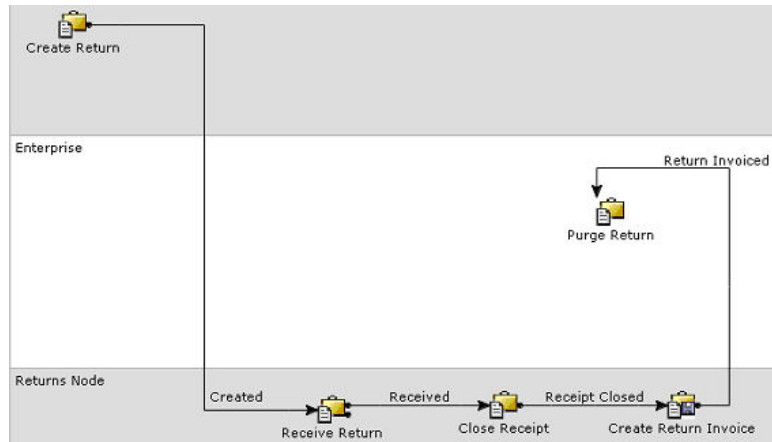
Based on the pipeline determination condition given below, the Blind RMA Pipeline is used for Blind RMA Return Order fulfillment.



Note: The condition "Is Blind RMA" mentioned in the figure is configured as OrderType='03'.

Return releases are not created for these return orders. However, a receipt is recorded against the Return Order.

The Blind RMA Pipeline should be configured according to the following pipeline:



Return Order Interface Data Mapping

RMAHDR - Return Release Download Header

The following table lists the header information required by the Return Release Download API.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
WHSE	OrderRelease/@ReceivingNode	5	1
RECORD-TYPE	"RMAHDR"	6	6
ACTION-CODE	"CH" or "CL" based on modification or closure	2	12
RMA-NUMBER	OrderRelease/Order/@OrderNo	15	14
RMA-RELEASE-NO	OrderRelease/@ReleaseNo	3	29
RMA-TYPE	OrderRelease/Order/@OrderType	2	32
EXPECTED-NO-OF-CASES	N/A	5	34
EXPECTED-NUMBER-OF-PALLETS	N/A	5	39
EXPECTED-NUMBER-OF-UNITS	N/A	7	44
TRAILER-NO	N/A	20	51
FREIGHT-COLLECT-FLAG	OrderRelease/Order/@TermsCode	1	71
EXPECTED-DATE	OrderRelease/Order/@OrderDate	8	72
CARRIER-CODE	OrderRelease/Order/@SCAC	4	80
INVOICE-NUMBER	N/A	20	84
SHIP-TO-CUST-ID	OrderRelease/OrderLine/@ShipToID	10	104
BILL-TO-CUST-ID	OrderRelease/Order/@BillToID	10	114
ENTRY-DATE	OrderRelease/Order/@OrderDate	8	124
SHIP-TO-NAME	OrderRelease/PersonInfoShipTo/ @FirstName + @LastName	25	132
BILL-TO-SHORT-NAME	OrderRelease/Order/ PersonInfoBillTo/@FirstName + @LastName	12	157

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
SHIP-TO-ADDR-1	OrderRelease/PersonInfoShipTo/ @AddressLine1	30	169
SHIP-TO-ADDR-2	OrderRelease/PersonInfoShipTo/ @AddressLine2	30	199
SHIP-TO-ADDR-3	OrderRelease/PersonInfoShipTo/ @AddressLine3	30	229
SHIP-TO-CITY	OrderRelease/PersonInfoShipTo/ @City	30	259
SHIP-TO-STATE-CODE	OrderRelease/PersonInfoShipTo/ @State	2	289
SHIP-TO-ZIP	OrderRelease/PersonInfoShipTo/ @ZipCode	9	291
SHIP-TO-COUNTRY-CODE	OrderRelease/PersonInfoShipTo/ @Country	5	300
CLAIM-NO	N/A	20	305
PICKTICKET-NO	N/A	20	325
REASON-CODE	N/A	4	345
PRO-NUMBER	N/A	20	349
REFERENCE-FIELD-1	OrderRelease/Order/ @EnterpriseCode	20	369
REFERENCE-FIELD-2	N/A	20	389
REFERENCE-FIELD-3	N/A	20	409
REFERENCE-FIELD-4	N/A	20	429
REFERENCE-FIELD-5	N/A	20	449
REFERENCE-FIELD-6	N/A	20	469
REFERENCE-FLAG-1	N/A	1	489
REFERENCE-FLAG-2	N/A	1	490
REFERENCE-FLAG-3	N/A	1	491
REFERENCE-FLAG-4	N/A	1	492
REFERENCE-FLAG-5	N/A	1	493
REFERENCE-FLAG-6	N/A	1	494
WMS-BUFFER	Defaulted with blank spaces	30	495

RMADTL - Return Release Download Detail

The following table lists the detail or line information required by the Return Release Download API.

DCS Parameter	Sterling Selling and Fulfillment Foundation Parameter	Field Size	Start Position
WHSE	OrderRelease/@Receiving Node	5	1
RECORD-TYPE	"RMADTL"	6	6
ACTION-CODE	"CH" or "CL" based on modification or closure	2	12
RMA-NUMBER	OrderRelease/Order/@OrderNo	15	14
RMA-RELEASE-NO	OrderRelease/@ReleaseNo	3	29
RMA-LINE-NO	OrderRelease/OrderLine/@PrimeLineNo	5	32
RMA-SUB-NO	Default value '0'	5	37
ITEM-ID	OrderRelease/OrderLine/Item/@ItemID	24	42
PRODUCT-CLASS	OrderRelease/OrderLine/Item/@ProductClass	2	66
QUALITY-STATUS	Defaulted in INTERFACE_DEFAULTS	2	68
PACK-TYPE	Defaulted in INTERFACE_DEFAULTS	4	70
EXPECTED-QUANTITY	OrderRelease/OrderLine/OrderStatuses/OrderStatus/@StatusQuantity	9	74
RMA-REASON-CODE	OrderRelease/OrderLine/@ReturnReason	4	83
DISPOSITION-CODE	N/A	2	87
CREDIT-FLAG	OrderRelease/Order/@TermsCode	1	89
PSEUDO-SERIAL-NUMBER	N/A	20	90
INVOICE-NUMBER	N/A	20	110
PICKTICKET-NO	N/A	20	130

RMACMT- Return Release Download Comments

The following table lists the comment information required by the Return Release Download API.

DCS Parameter	Release 5.0 Parameter	Field Size	Start Position
WHSE	OrderRelease/@Receiving Node	5	1
RECORD-TYPE	"RMACMT"	6	6
ACTION-CODE	"CH" or "CL" based on modification or closure	2	12
RMA-NUMBER	OrderRelease/Order/@OrderNo	15	14
RMA-RELEASE-NO	OrderRelease/@ReleaseNo	3	29
RMA-LINE-NO	OrderRelease/OrderLine/@PrimeLineNo or '0' for header level comment	5	32
RMA-SUB-NO	Default value '0'	5	37

DCS Parameter	Release 5.0 Parameter	Field Size	Start Position
COMMENT-SEQ-NO	OrderRelease/Order/Instructions/ Instruction/@SequenceNo OR OrderRelease/Orderline/Instructions/ Instruction/@SequenceNo	5	42
COMMENT-TYPE	Maps to appropriate DCS Comment Type	2	47
COMMENT-TEXT	OrderLine-> InstructionText OR Order -> InstructionText- '0' as return line number	80	49

Return Receipt Upload Data Mapping

Data Mapping to Create Return Order for Blind Return:

The following table lists the interface attribute mapping to create return orders if they do not exist in Sterling Selling and Fulfillment Foundation.

Sterling Selling and Fulfillment Foundation	DCS
Order/DocumentType	Default value for Return Document '0003'
Order/OrderDate	RARHDR.RECEIVED-DATE
Order/OrderNo	RARHDR.RMA_NUMBER
Order/OrderType	Default Value '03'
Order/SCAC	RARHDR. CARRIER-CODE
Order/TermsCode	RARHDR. FREIGHT-COLLECT
Order/PersonInfoShipTo/ FirstName	RARHDR. CUSTOMER-NAME
Order/PersonInfoShipTo/ AddressLine1	RARHDR. ADDRESS-1
Order/PersonInfoShipTo/ AddressLine2	RARHDR. ADDRESS-2
Order/PersonInfoShipTo/ AddressLine3	RARHDR. ADDRESS-3
Order/PersonInfoShipTo/ City	RARHDR. CITY
Order/PersonInfoShipTo/ State	RARHDR. STATE
Order/PersonInfoShipTo/ ZipCode	RARHDR. ZIP

Sterling Selling and Fulfillment Foundation	DCS
Order/PersonInfoShipTo/ Country	RARHDR. COUNTRY-CODE
Order/PersonInfoBillTo/ FirstName	RARHDR. CUSTOMER-NAME
Order/PersonInfoBillTo/ AddressLine1	RARHDR. ADDRESS-1
Order/PersonInfoBillTo/ AddressLine2	RARHDR. ADDRESS-2
Order/PersonInfoBillTo/ AddressLine3	RARHDR. ADDRESS-3
Order/PersonInfoBillTo/ City	RARHDR. CITY
Order/PersonInfoBillTo/ State	RARHDR. STATE
Order/PersonInfoBillTo/ ZipCode	RARHDR. ZIP
Order/PersonInfoBillTo/ Country	RARHDR. COUNTRY-CODE
Order/EnterpriseCode	RARHDR.REFERENCE-1, if the value of RARHDR.REFERENCE-1 is a valid Organization with Enterprise role. If the value of RARHDR.REFERENCE-1 is blank, this becomes the primary enterprise of the receiving node's organization. If the value of RARHDR.REFERENCE-1 is not a valid enterprise code, the system throws an error.
OrderLine/ReceivingNode	RARHDR.WHSE
Order/Instructions/ InstructionText	RARCMT.COMMENT-TEXT (if RARCMT.RMA-LINE-NO=0)
Order/Instructions/ InstructionType	RARCMT.COMMENT-TYPE (if RARCMT.RMA-LINE-NO=0)
Order/Instructions/ SequenceNo	RARCMT.SEQ_NUMBER (if RARCMT.RMA-LINE-NO=0)
OrderLine/PrimeLineNo	RARDTL.RMA-LINE-NO
OrderLine/OrderedQuantity	RARDTL.QUANTITY
OrderLine/Item/ItemID	RARDTL.ITEM_ID
OrderLine/Item/ProductClass	RARDTL.PRODUCT_CLASS
OrderLine/SubLineNo	Default Value '0'
OrderLine/Item/UnitofMeasure	Default Value 'EACH'

Sterling Selling and Fulfillment Foundation	DCS
OrderLine/ReturnReason	RARDTL.RMA-REASON-CODE
OrderLine/Instructions/Instruction/InstructionText	RARCMT.COMMENT-TEXT
OrderLine/Instructions/Instruction/InstructionType	RARCMT.COMMENT-TEXT
OrderLine/Instructions/Instruction/SequenceNo	RARCMT.SEQ-NUMBER

Data Mapping to Record Return Receipts:

The following table lists the interface attribute mapping to record return receipts on Sterling Selling and Fulfillment Foundation.

Table 7. Data Mapping to Record Return Receipts

Sterling Selling and Fulfillment Foundation	DCS
Receipt/ReceiptNo	RARHDR.WORKSHEET-NO
Receipt/EnterpriseCode	RARHDR.REFERENCE-1 if the receipt is not against a blind RMA. Otherwise the enterprise code is same as that of the blind RMA.
Receipt/ReleaseNo	RARHDR.RMA-RELEASE-NO
ReceiptLine/InspectedBy	RARDTL.USERID
ReceiptLine/InspectionComments	RARDTL.RMA-REASON-CODE
ReceiptLine/DispositionCode	RARDTL.DISPOSITION-CODE
ReceiptLine/InspectionDate	RARHDR.RECEIVED-DATE
Receipt/OrderNo	RARDTL.RMA-NUMBER
ReceiptLine/PrimeLineNo	RARDTL.RMA-LINE-NO
ReceiptLine/Quantity	RARDTL.QUANTITY
ReceiptLine/SerialNo	RARDTL.SERIAL-NO
ReceiptLine/SubLineNo	Default Value '1'

Assumptions and Limitations

The assumptions and limitations in the integration of Sterling Selling and Fulfillment Foundation with DCS for returns interface are listed below:

- The integration to DCS is at return release rather than return creation. This is done to support returns that may require a manual credit check or approval before it is accepted (released).
To send a return order to DCS whenever a return is created, you can model a service to call Return Release upon creation, based on a return type.
- The Return Order number in Sterling Selling and Fulfillment Foundation is unique across all enterprises.
- All Return Order lines must use consecutive prime line numbers, with all sub line numbers as '0'. The RMA record always sets the RMA_SUB_NO as '0'.
- Only one release is supported for each receiving node of the Return Order. To apply this, enable the document type level rule 'Consolidate New Releases' for

the 'Reverse Logistics' document type. This allows the new lines added to the Return Order to be included in the existing release.

- Receipt is allowed only for items included in the Return Order. To receive an item that is not in the return, a line with that item should be added into the return release and downloaded into DCS again.
- Inventory updates during return receipt upload should be turned off. Inventory adjustments for return receipts should be done through the inventory adjustment interface with DCS. Whenever inventory is updated in DCS, the inventory is updated in Sterling Selling and Fulfillment Foundation too through this interface.
- The following modifications are allowed on a Return Order:
 - Order Level
 - ADD_LINE: A new line can be added to the Return Order. This line is added in the created status. Based on the 'Consolidate New Releases' setting in the 'Reverse Logistics' document type level, this new line is added into the existing release during the release process and the entire release is downloaded to DCS.
 - Order Line Level
 - Modifications are not allowed in the Return Order line level.
 - Order Release Level
 - ADD_LINE: A new release line can be added.
 - CANCEL: IBM recommends that you disallow cancellation once the return release is sent to DCS. This is because the return receipt upload agent throws an exception if the return is being received or has already been received in DCS while it is getting cancelled on Sterling Selling and Fulfillment Foundation.
 - ADD_QUANTITY: A release line quantity can be added.
 - The other modifications allowed are Add Note, Change BillTo, Change Carrier, Change Carrier Account No, Change Carrier Service Code, Change Freight Terms, Change Delivery Code, Change MarkFor, Change ReqShipDate, and Change ShipTo.
- Receipt overage is not allowed in DCS. A new return line must be created on Sterling Selling and Fulfillment Foundation and downloaded to DCS upon release.
- Return Orders with Kit items should be created as blind returns on DCS. They cannot be created for sales orders in Sterling Selling and Fulfillment Foundation.
- Return Orders with Kit items should contain return lines for kit components.
- Return Orders can be created for multiple sales orders and can be received in DCS.
- The return receipt upload agent does not upload instructions to Sterling Selling and Fulfillment Foundation if the instruction text is blank.
- The configuration assumptions for DCS are:
 - The creation of a return in DCS is enabled only for blind returns.
 - For blind returns on DCS, a new function should be configured to "Create Blind RMA" with RMA_Type='03' defaulted and protected. This ensures that blind RMAs are always created with RMA_Type = '03'.
 - The ability to receive an overage item or a different item on a return is disabled.

For more information about configuring DCS Inventory updates, see the *Yantra 5x Configuration Guide*.

Chapter 3. Integrating with Stand-Alone Sterling Warehouse Management System

Limitation on Use of Services with Stand-Alone Sterling Warehouse Management System

The use of services with the stand-alone Sterling Warehouse Management System are not supported in a multischema environment.

Installing Integration Pack for Receipt and Inventory Change Upload Interfaces on a Sterling Warehouse Management System Instance

About this task

To install the receipt and inventory change upload components on the Sterling Warehouse Management System instance:

Procedure

1. Set the environment variable `INSTALL_DIR` to point to the Sterling Selling and Fulfillment Foundation installation directory.
2. Change the directory to `<INSTALL_DIR>/bin`, and run the following command for UNIX or Linux:

```
sci_ant.sh -f wms_integration_pack_installer.xml
```

 (or `sci_ant.cmd` for Windows).
3. After you run the above command, check the contents of the `wms_integration_pack_fc_installer.xml.restart` file located in the `<INSTALL_DIR>/database/FactorySetup/install/` directory. In the `wms_integration_pack_fc_installer.xml.restart` file make sure that the "Completed" attribute of the TaskInfo element is set to "Y". If this is set to "N", fix the integration pack installation problems, and repeat the previous step.

Installing Integration Pack for Receipt and Inventory Change Upload Interfaces on a Sterling Distributed Order Management Instance

About this task

Note: If your Sterling Distributed Order Management instance is on a release that is prior to Release 9.0, you must copy the following files located in the runtime directory of the Sterling Warehouse Management System instance to the runtime directory of the Sterling Distributed Order Management instance.

- `<INSTALL_DIR>/bin/omp_integration_pack_installer.xml`
- `<INSTALL_DIR>/database/FactorySetup/install/omp_integration_pack_fc_installer.xml`
- `<INSTALL_DIR>/database/FactorySetup/IntegrationPack/IP_OMP_*.xml`

To install the receipt and inventory change upload components on the Sterling Distributed Order Management instance:

Procedure

1. Set the environment variable `INSTALL_DIR` to point to the Sterling Selling and Fulfillment Foundation installation directory.
2. Change the directory to `<INSTALL_DIR>/bin`, and run the following command for UNIX or Linux:

```
sci_ant.sh -f omp_integration_pack_installer.xml
```

 (or `sci_ant.cmd` for Windows).
3. After you run the above command, check the contents of the `omp_integration_pack_fc_installer.xml.restart` file located in the `<INSTALL_DIR>/database/FactorySetup/install/` directory. In the `omp_integration_pack_fc_installer.xml.restart` file make sure that the "Completed" attribute of the TaskInfo element is set to "Y". If this is set to "N", fix the integration pack installation problems, and repeat the previous step.

Uploading Receipts

Sterling Selling and Fulfillment Foundation supports integration between Sterling Distributed Order Management and Sterling Warehouse Management System for uploading receipts and receipt adjustments. To integrate Sterling Distributed Order Management and Sterling Warehouse Management System, you must configure a common JMS queue. You must also model the node on both instances. For the Sterling Distributed Order Management instance, model the node as a non-Sterling Warehouse Management System integrated node.

Uploading receipts has the following integration touch points:

- Sterling Warehouse Management System - Uploading Receipt Information and Uploading Receipt Adjustment Information
- DOM Components - Loading Receipt Information from a Node and Loading Receipt Adjustment Information from a Node

Uploading Receipt Information

To upload the receipt details from Sterling Warehouse Management System to Sterling Distributed Order Management, use the ReceiptUpload-751 service.

ReceiptUpload-751 Service

This service is invoked from the WMS instance.

The receiveOrder API is invoked during the receiving process. When the receiving process for a case or pallet is complete, and the user closes the case or pallet, or when receiving for a loose SKU is complete, one of the `ON_CASE_RECEIPT`, `ON_PALLET_RECEIPT`, and `ON_SKU_RECEIPT` events of the `RECEIVE_RECEIPT` transaction is raised. To invoke the ReceiptUpload-751 service, ensure that the UploadReceipt action, under `Order>PO Order Receipt>Actions>Receipt Upload`, is configured on these events.

The ReceiptUpload-751 service then translates the API output and serves as an input to the receiveOrder API. This is published as a message to the JMS queue of the web server of the DOM instance.

This service invokes the getReceiptLinesList API. The getReceiptLinesList API has been modified to use an additional flag called RelevantItemLinesOnly. If this flag is set to "Y", the API returns the relevant lines exploding the hierarchical information of LPNs as necessary, satisfying the input criteria.

This flag is relevant if:

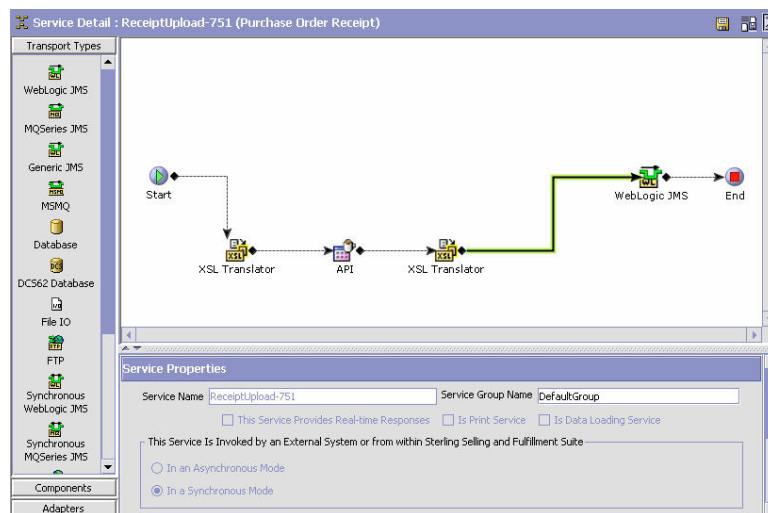
- Either the case identifier or pallet identifier is passed as input.
- The case identifier or pallet identifier passed is not shipped out of the warehouse.

Configuring the ReceiptUpload-751 Service About this task

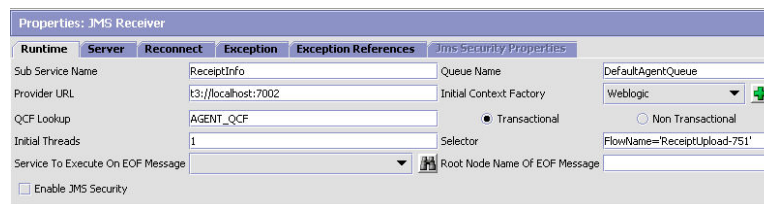
To configure the ReceiptUpload-751 service:

Procedure

1. From the Applications menu of the Applications Manager, select Application Platform.
2. From the tree in the application rules side panel, double-click Process Modeling.
3. Click the Order tab. In the Process Types swimlane, right-click the Purchase Order Receipt process type and click Model Process. The Repository Details window and work area are displayed for the Order process type.
4. Click the Service Definitions tab.
5. Expand the DefaultGroup branch.
6. Right-click ReceiptUpload-751 and select Details. The Service Detail window appears in the work area.



7. Click the green connector that connects the XSL Translator and the WebLogic JMS. The JMS Sender properties displays as shown.



8. In the Runtime tab, make sure that the "Commit of this message depends on parent transaction" box is checked.

For field value descriptions, refer to the Service Builder Nodes and Parameters appendix of the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Uploading Receipt Adjustment Information

To upload receipt adjustment details from Sterling Warehouse Management System to Sterling Distributed Order Management, use the AdjustReceiptUpload-751 service.

AdjustReceiptUpload-751 Service

This service is invoked from the WMS instance.

The unreceiveOrder API is invoked during the unreceiving process. When the unreceiving process is complete, the ON_SUCCESS event of the UNRECEIVE_RECEIPT transaction is raised. To invoke the AdjustReceiptUpload-751 service, ensure that the adjustReceiptUpload action, under Order>PO Order Receipt>Actions>Receipt Upload, is configured on the ON_SUCCESS event.

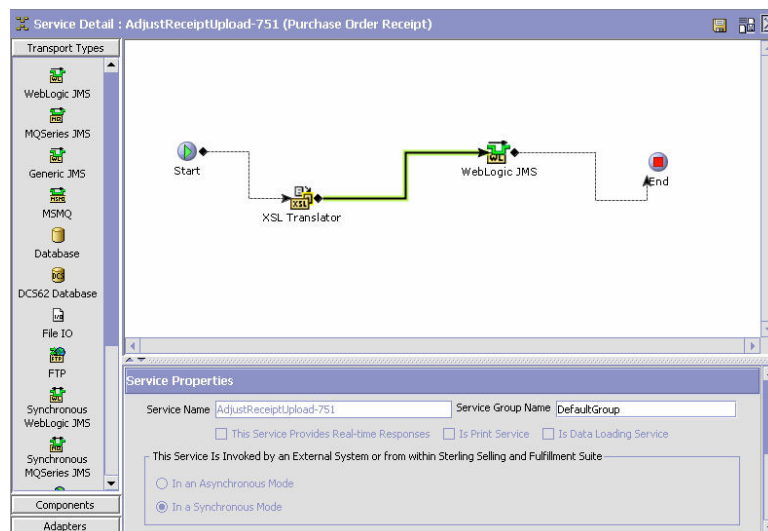
The AdjustReceiptUpload-751 service then translates the API output and serves as an input to the unreceiveOrder API. This is published as a message in the JMS queue of the web server of the Sterling Distributed Order Management instance.

Configuring Updated Receipt Adjustment Information from a Node: About this task

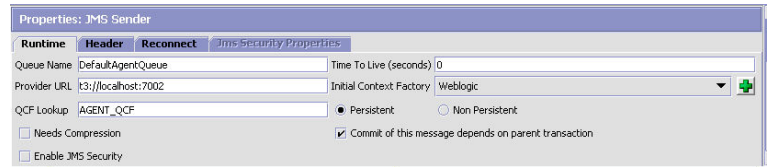
To configure the AdjustReceiptUpload-751 service:

Procedure

1. From the Applications menu of the Applications Manager, select Application Platform.
2. From the tree in the application rules side panel, double-click Process Modeling.
3. Click the Order tab. In the Process Types swimlane, right-click the Purchase Order Receipt process type and click Model Process. The Repository Details window and work area are displayed for the Order process type.
4. Click the Service Definitions tab.
5. Expand the DefaultGroup branch.
6. Right-click AdjustReceiptUpload-751 and select Details. The Service Detail window appears in the work area.



- Click the green connector that connects the XSL Translator and the WebLogic JMS. The JMS Sender properties displays as shown.



- In the Runtime tab, make sure that the "Commit of this message depends on parent transaction" box is checked.
For field value descriptions, refer to the Service Builder Nodes and Parameters appendix of the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Loading Receipt Information from a Node

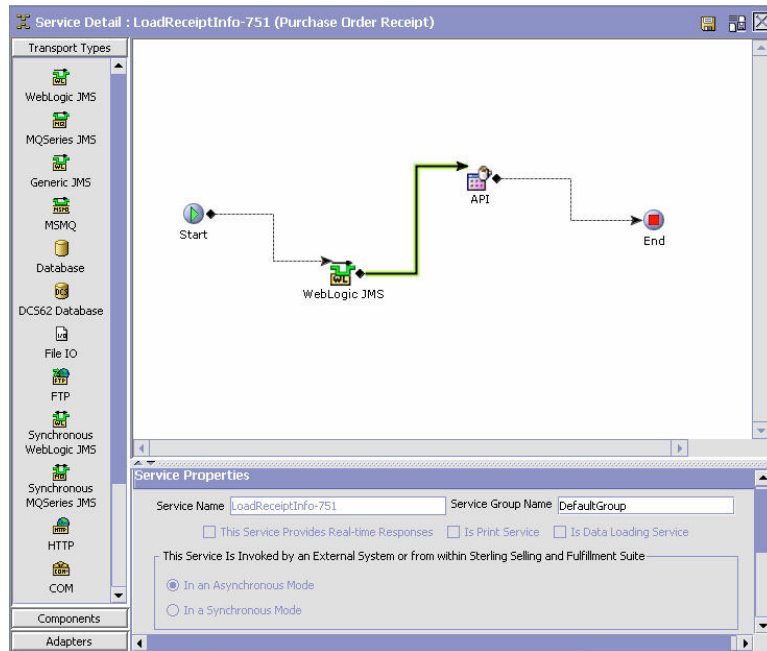
About this task

The LoadReceiptInfo-751 service is used at the Sterling Distributed Order Management instance to retrieve receipt details from the node.

To retrieve receipt details, set up the LoadReceiptInfo-751 service for Sterling Distributed Order Management instance.

Procedure

- From the Applications menu of the Applications Manager, select Application Platform.
- From the tree in the application rules side panel, double-click Process Modeling.
- Click the Order tab. In the Process Types swimlane, right-click the Purchase Order Receipt process type and click Model Process. The Repository Details window and work area are displayed for the Order process type.
- Click the Service Definitions tab.
- Expand the DefaultGroup branch.
- Right-click LoadReceiptInfo-751 and select Details. The Service Detail window appears in the work area.



LoadReceiptInfo-751 Service

This service is invoked from the Sterling Distributed Order Management instance.

Although we have used Weblogic JMS as an example, the Sterling Selling and Fulfillment Foundation also supports the use of IBM WebSphere® and JBoss Messaging JMS.

Configuring the LoadReceiptInfo-751 Service: About this task

The LoadReceiptInfo-751 service reads the message from the JMS queue and invokes the receiveOrder API.

To configure the service:

Procedure

In the Service Detail: LoadReceiptInfo-751 window, click the green connector that connects the WebLogic JMS and the API. The JMS Receiver properties displays as shown.

For field value descriptions of the fields, refer to the Service Builder Nodes and Parameters appendix of the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Loading Receipt Adjustment Information from a Node

About this task

The LoadReceiptAdjustments-751 service is used at the Sterling Distributed Order Management instance to retrieve receipt details from the node.

To retrieve receipt details:

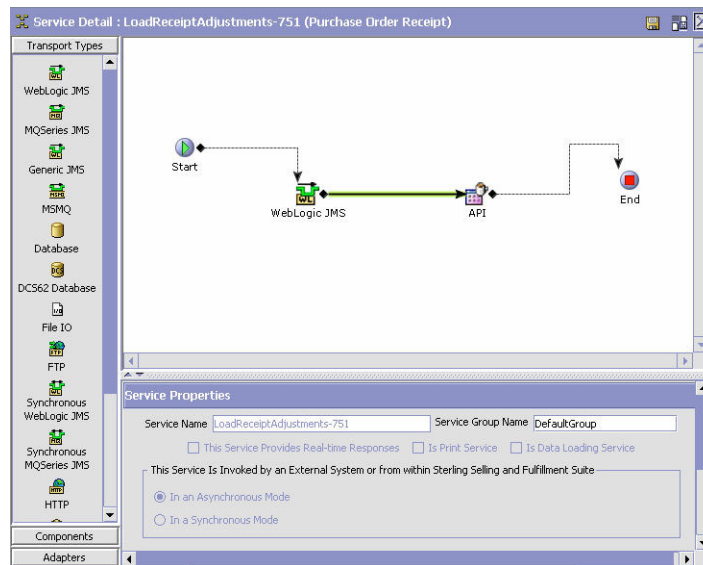
Procedure

1. From the Applications menu of the Applications Manager, select Application Platform.
2. From the tree in the application rules side panel, double-click Process Modeling.
3. Click the Order tab. In the Process Types swimlane, right-click the Purchase Order Receipt process type and click Model Process.

The Repository Details window and work area are displayed for the Order process type.

4. Click the Service Definitions tab.
5. Expand the DefaultGroup branch.
6. Right-click LoadReceiptAdjustments-751 and select Details.

The Service Detail window appears in the work area.



LoadReceiptAdjustments-751 Service

This service is invoked from the Sterling Distributed Order Management instance.

Although we have used WebLogic JMS as an example, Sterling Selling and Fulfillment Foundation supports the use of IBM WebSphere and JBoss Messaging JMS.

The LoadReceiptAdjustments-751 service reads the message from the JMS queue and invokes the unreceiveOrder API.

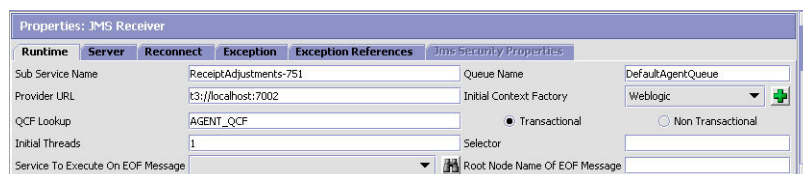
Configuring the LoadReceiptAdjustments-751 Service:

About this task

To configure the service:

Procedure

In the Service Detail: LoadReceiptAdjustments-751 window, click the green connector that connects the WebLogic JMS and the API. The JMS Receiver properties displays as shown.



For field value descriptions, refer to the Service Builder Nodes and Parameters appendix of the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Uploading Inventory Changes at a Node

Sterling Selling and Fulfillment Foundation provides inventory integration between Sterling Distributed Order Management and Sterling Warehouse Management System that are running on two different instances. To synchronize inventory between separate Sterling Distributed Order Management and Sterling Warehouse Management System instances, you must configure a common JMS queue. You must also model the node on both instances. For a Sterling Distributed Order Management instance, model the node as a non-Sterling Warehouse Management System integrated node.

The uploading process is performed in two phases, which are described in the following topics.

Uploading Updated Inventory Information

To keep inventory information between Sterling Distributed Order Management and Sterling Warehouse Management System instances in synchronization, use the InventoryChangeUpload-751 service.

InventoryChangeUpload-751 Service

Inventory information needs to be transmitted to the Sterling Distributed Order Management instance for all adjustment types other than RECEIPT, RETURN, and SHIPMENT. (Inventory for these adjustment types would typically be transmitted by means of receipt or shipping interfaces). The InventoryChangeUpload-751 service is invoked from the WMS instance on the SUPPLY_CHANGE event of the INVENTORY_CHANGE transaction, which is raised whenever inventory changes at a node. To invoke the InventoryChangeUpload-751 service, ensure that the UploadInventoryChange action, under General>General>Actions>Inventory Synchronization, is configured on the SUPPLY_CHANGE event.

This service then translates the output of the SUPPLY_CHANGE event and creates an input XML for the adjustInventory API. This input XML is published as a message to the JMS queue of the web server of the DOM instance.

The "doesAdjustmentTypeRequiresTransmission" condition is used to determine which inventory changes require transmission. This condition returns true if the adjustment type is any value other than RECEIPT, RETURN, and SHIPMENT.

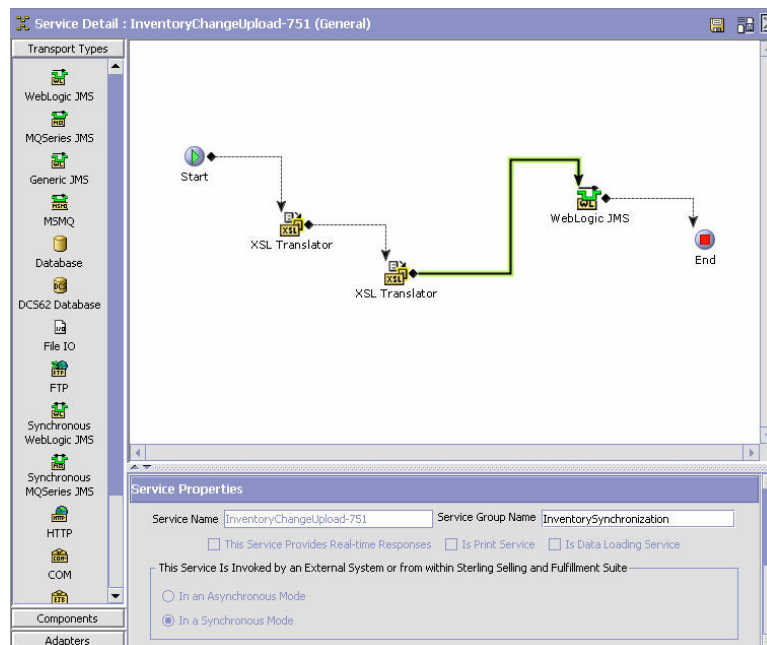
Configuring the Updated Inventory Information from a Node

About this task

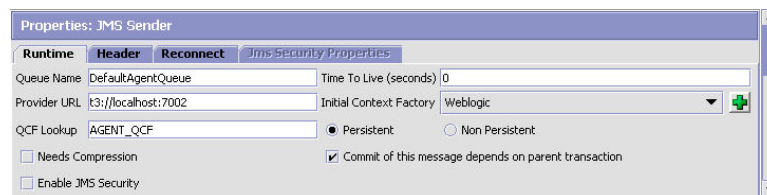
To configure the service:

Procedure

1. From the Applications menu of the Applications Manager, select Application Platform.
2. From the tree in the application rules side panel, double-click Process Modeling.
3. Click the General tab. In the Process Types swimlane, right-click the General process type and select Model Process. The Repository Details window and work area displays for the General process type.
4. Click the Service Definitions tab.
5. Expand the InventorySynchronization branch.
6. Right-click InventoryChangeUpload-751 and select Details. The Service Detail window appears in the work area.



7. Click the green connector that connects the XSL Translator and the WebLogic JMS. The JMS Sender properties displays as shown.



8. In the Runtime tab, make sure that the "Commit of this message depends on parent transaction" box is checked.

For field value descriptions, refer to the Service Builder Nodes and Parameters appendix of the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Loading Inventory Information from a Node

About this task

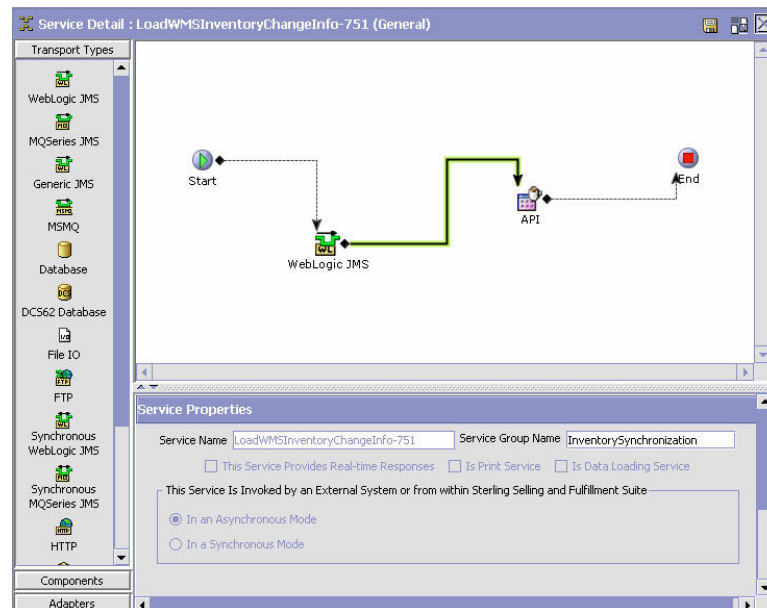
In order to reconcile the inventory picture between Sterling Distributed Order Management and Sterling Warehouse Management System, the inventory picture at the Sterling Warehouse Management System instance must be loaded to the Sterling Distributed Order Management instance.

To reconcile the inventory picture, set up the LoadWMSInventoryChangeInfo-751 service for the Sterling Distributed Order Management instance:

Procedure

1. From the Applications menu of the Applications Manager, select Application Platform.
2. From the tree in the application rules side panel, double-click Process Modeling.
3. Click the General tab. In the Process Types swimlane, right-click the General process type and select Model Process. The Repository Details window and work area displays for the General process type.
4. Click the Service Definitions tab.
5. Expand the InventorySynchronization branch.
6. Right-click LoadWMSInventoryChangeInfo-751 and select details.

The Service Detail window appears in the work area.



LoadWMSInventoryChangeInfo-751 Service

This service is invoked from the Sterling Distributed Order Management instance.

Although we have used WebLogic JMS as an example, Sterling Selling and Fulfillment Foundation also supports the use of IBM WebSphere and JBoss Messaging JMS.

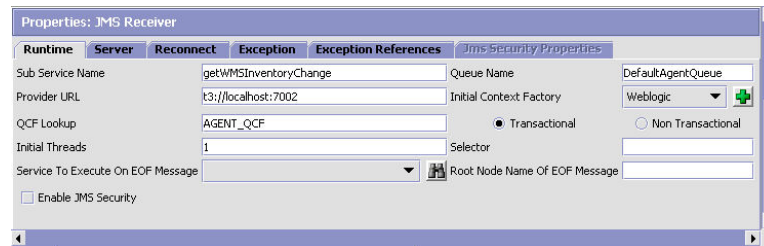
The LoadWMSInventoryChangeInfo-751 service reads the message from the JMS queue and invokes the adjustInventory API.

Configuring the LoadWMSInventoryChangeInfo-751 Service: About this task

To configure the service:

Procedure

In the Service Detail: LoadWMSInventoryChangeInfo-751 window, click the green connector that connects the WebLogic JMS and API. The JMS Receiver properties displays as shown.



For field value descriptions, refer to the Service Builder Nodes and Parameters appendix of the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Uploading Inventory Snapshots

Sterling Selling and Fulfillment Foundation provides the ability to upload inventory snapshots for integrating Sterling Warehouse Management System and Sterling Distributed Order Management that are running on different instances. This involves loading the inventory picture from a Sterling Warehouse Management System instance to a Sterling Distributed Order Management instance.

Generating Inventory Snapshot Files

About this task

A single XML file is generated by running the inventory snapshot component at a WMS instance where data is fetched from the YFS_Inv_SnapShot_VW view. This view is derived from the following tables:

- YFS_INVENTORY_ITEM
- YFS_INVENTORY_SUPPLY
- YFS_INVENTORY_TAG

To run the inventory snapshot component on a Sterling Warehouse Management System instance:

Procedure

1. Go to the <INSTALL_DIR>/bin directory.
2. For UNIX or Linux, run this command:

```
sci_ant.sh -f runInventorySnapShot.xml -DFilePath=<FilePath>  
-DShipNode=<ShipNode> -DReasonCode=<ReasonCode> -DReasonText=<ReasonText>  
-DItemsPerGroup=<ItemsPerGroup>
```

For Windows, run this command:

```
sci_ant.cmd -f runInventorySnapShot.xml -DFilePath=<FilePath>  
-DShipNode=<ShipNode> -DReasonCode=<ReasonCode> -DReasonText=<ReasonText>  
-DItemsPerGroup=<ItemsPerGroup>
```

These are the parameters passed for the inventory snapshot:

Table 8. Inventory Snapshot

Field	Description
FilePath	The absolute path of the directory where the generated XML file is stored.
ShipNode	The ship node for which the XML file is generated.
ReasonCode	The reason code that is defined by the user.
Reason Text	The reason code text that is that is defined by the user.
ItemsPerGroup	The number of item tags in the items tag element. The recommended value is 100. However, you could specify any value from 1 to 100.

Results

Note: The time taken to generate an XML file on a WMS instance is not more than 3 minutes when the number of records in the YFS_INVENTORY_SUPPLY table are 430,000 and 512 M heap is used.

These generated XML files can be shared by both Sterling Warehouse Management System and Sterling Distributed Order Management instances through NFS mounts or can also be transferred through FTP to the DOM instance.

For more information about uploading inventory snapshot components on a Sterling Distributed Order Management instance, refer to the *Sterling Selling and Fulfillment Foundation: Global Inventory Visibility Configuration Guide*.

Chapter 4. Integrating with Third-Party Warehouse Management Systems

Introduction to Third-Party Warehouse Management System Integration

Sterling Selling and Fulfillment Foundation enables you to integrate with external third-party warehouse management systems in order to identify external ship nodes, manage external inventory and distribution of items, and coordinate external warehouse activities.

The Sterling Selling and Fulfillment Foundation provides complete functionality for Distributed Order Management and Warehouse Management systems without the need for integration. For more information about the IBM Sterling Warehouse Management System, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Concepts Guide*.

Third-Party Warehouse Management Systems

Sterling Selling and Fulfillment Foundation provides XML-based integration to third-party warehouse management systems (WMS). To integrate Sterling Selling and Fulfillment Foundation with third-party warehouse management systems, configure them using services, as indicated in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*. In addition, use the following APIs when necessary:

- `getUnprocessedImportDataEx()` – Retrieves unprocessed data from import tables.

Third-Party Shipment Advice

When creating shipment advice data for third-party software, use services to stage your data. For more information about using services, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Third-Party Inventory Change

Sterling Selling and Fulfillment Foundation enables XML-based integration with third-party warehouse management inventory control systems through services or through system APIs. The following APIs enable integration with third-party systems for inventory change:

- `getInventorySnapShot` – Obtains total number of items in inventory at all ship nodes.
- `getInventoryMismatch` – Detects or corrects mismatches between the global inventory picture on Sterling Selling and Fulfillment Foundation and the global inventory picture on the external system.
- `adjustInventory` – Applies corrections to the global inventory picture in Sterling Selling and Fulfillment Foundation. This could also be used to correct a mismatch when the `getInventoryMismatch` API is used to detect the mismatches.

Chapter 5. Integrating with the Loftware Print Server and Label Manager

Overview of Loftware Print Server and Label Manager Integration

The Sterling Warehouse Management System can be integrated with the Loftware Print Server (LPS) and Loftware Label Manager (LLM), and custom prints can be configured.

For more information about installing and configuring the Loftware Print Server, see the *Sterling Selling and Fulfillment Foundation: Installation Guide*.

For more information about server requirements and installation guidelines of Loftware Label Manager, see the *Loftware Print Server User's Guide* and *Loftware Label Manager User's Guide*.

For more information about configuring printers, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Configuration Guide*.

Sterling Warehouse Management System supports all capabilities of the Loftware Printer Server, including cluster installations. For specific details, consult <http://www.loftware.com>.

Standard Labels Provided by the Sterling WMS

The Sterling Warehouse Management System provides the following Standard Labels:

- UCC 128 Container Shipping Label
- VICS Bill Of Lading
- Packing Slip
- Batch Sheets
 - Item Pick Batch Sheet
 - Cart Manifest Batch sheet
- Count Sheet
- UPS Carrier Label
- UPS Pickup Summary

Important Assumptions About Standard Labels

The factory shipped Cart Manifest Print is based on the following assumptions:

1. Each cart location is assumed to have 1 slot, when the number of locations in the cart is greater than 8.
2. Each cart location is assumed to have 2 slots, when the number of locations in the cart is less than or equal to 8.

For example, if the cart locations in the cart are named as A, B, C, ... H, then the Cart Manifest Print has locations such as A1, A2, B1, B2, C1, C2, ... H1, H2.

Thus, the task type "Number of containers allowed per location in the equipment" should always be set at 1 or 2.

For other configurations of the Cart, the Print has to be customized.

Printing Standard Labels

To print these standard labels, the Sterling Warehouse Management System provides services associated with events. By default, the events are disabled. Enable the events if you want to print the standard labels. The following table lists the services provided in the Sterling Warehouse Management System.

Service Name	Event	Description
PrintShippingLabel	ADD_TO_CONTAINER.ON_CONTAINER_PACK_COMPLETE	Prints a UCC-128 Shipping Label for a container
PrintShipmentContainerLabels	Reprint Request from console	Prints UCC-128 Container Labels for Containers in the Shipment
PrintShipmentBOL	CONFIRM_SHIPMENT.ON_SUCCESS	Print a VICS BOL for Shipment
PrintTaskList	Reprint Request from console	Prints a BatchSheet (CartManifest or ItemPickBatch Sheet) or a CountSheet, based on the ActivityGroup for the Batch. If the Batch belongs to the ActivityGroup COUNT, the CountSheet is printed.
PrintLoadBOL	RECEIVE_IN_TRANSIT_UPDATES.ON_SUCCESS	Prints a VICS BOL for Load
PrintWave	PRINT_WAVE.ON_SUCCESS	Prints PickList (BatchSheets), Container Labels and pre-generates PackLists for Shipments in the Wave
PrintPackList	ADD_TO_CONTAINER.ON_SHIPMENT_PACK_COMPLETE	Prints a PackList
PickListPrint	PRINT_PICKLIST.ON_SUCCESS	Prints PackLists for Shipments in the PickList
PrintTaskSheets	COMPLETE_TASK.TASK_COMPLETED	Creates a Batch for successor Tasks of the completed task and Prints a BatchSheet for the same
PrintMoveTickets	RELEASE_MOVE_REQUEST.ON_SUCCESS	Creates a Batch for the MoveRequest and prints a BatchSheet for the same
PrintPostPickContainers	POST_PICK_CONTAINERIZATION.ON_SUCCESS	Prints UCC-128 Shipping Labels for containers created as part of Post Pick Containerization

Designing Custom Labels

Use Software Label Manager to design a label (creates an .lwl file). For more information about creating new labels using Software Label Manager, see *Software Label Manager User's Guide*.

Note: The Sterling Warehouse Management System requires the repeating fields in a label to have names in the format of <fieldname>_<integer>. The integer in the field name takes values like 1, 2, 3.

The Software Label Manager may be installed on any compatible PC. For more information about server requirements and installation guidelines, see *Software Print Server User's Guide*.

Note: While designing a custom label, use of the .LST file maintains uniformity in label field names across different labels. For more information about LST file(s), see *Software Label Manager User's Guide*.

Displaying Page Numbers

Procedure

To display Page Numbers and Total Number of Pages in the print output: add the following fields to the Label (.lwl file):

- PageNo
- TotalPages

Results

This ensures that the page numbers are displayed in the format Page X of N.

File Naming Conventions for Custom Labels

The Sterling Warehouse Management System requires the following naming conventions be followed while creating labels (.lwl files) using Software Label Manager:

- The first page of the label file created should be named in the format <filename>.lwl
- The middle page of the label file created should be named in the format <filename>_Mid.lwl
- The last page of the label file created should be named in the format <filename>_Last.lwl

File Design Conventions for Labels

The first page of the label and the last page of the label are always single pages. The middle page, on the other hand, is used n number of times in accordance with the total number of label pages to be printed.

For example, if a label print is six pages, the first page and last page make two pages, and the middle page (<filename>_Mid.lwl) is repeated four times.

Creating a New Label Format

You can print a label in single-page or multi-page format depending on the number of lines in the label. If the number of lines can be accommodated on the first page itself, you can print the label in single-page format. For this, you must

create a new label format (<filename>_SinglePage.lwl). For more information about creating a label format, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

After you create the new label format, the print service calls the xsl file to check the number of lines in the label. Depending on the number of lines, a single-page or multi-page label is printed. For example, the LTL Manifest Label can be printed in single-page or multi-page format.

Copying the Custom Label

About this task

After you create the custom label:

Procedure

Copy the label to the Runtime > Template > Label > Extn directory.

Generating a Mapping XML File for a Label

About this task

The GenLabelMappingXML.java tool is used to generate Mapping XML for a label designed using Software Label Manager. The output XML contains all the field names of the label. XPath bindings for the label fields have to be specified.

To generate a Mapping XML for a label:

Procedure

1. Invoke the GenLabelMappingXML tool:

```
java -classpath <classpath>  
com.yantra.tools.labelxmlmapping.GenLabelMappingXML  
<parameter1> <parameter2>
```

<parameter1> - File name of the .tab file generated when the label (.lwl) file is saved in Software Label Manager.

The full path, excluding the extension should be specified.

<parameter2> - File name of the XML file generated by the tool.

The full path, excluding the extension should be specified.

For example, to generate a Mapping XML for the label BOL.lwl, the .tab file name is BOL.tab

Example:

```
java -classpath  
platform_afc.jar;log4j-1.2.12.jar;xercesImpl.jar  
com.yantra.tools.labelxmlmapping.GenLabelMappingXML  
<path-of-the-file>/BOL <path-of-file>/BOLMap
```

2. Ensure that the classpath has the following jar files:

- platform_afc.jar
- log4j-1.2.12.jar
- xercesImpl.jar

3. Edit the map file (XML) generated for a label (LWL) to associate the XML data to the fields required on the label and copy it into the Sterling Selling and Fulfillment Foundation Runtime Template folder. See *Editing and Relocating the Map File Generated for a Label*.

XML File Settings Generated by GenLabelMappingXML.java

In the Mapping XML file generated using the GenLabelMappingXML.java tool:

- Each Label Field has a corresponding LabelField element
- Label Fields which are repeating are present in the RepeatingField element.
- Each of the Repeating Fields has a MaxFirstPage, MaxMidPage, and MaxLastPage, which denote the number of times the field is repeated in the First page, Middle Pages, and Last Page respectively.
- To repeat the same set of values of the field in all the pages, the RepeatValuesOnEachPage attribute should be set to "Y" in the RepeatingField element.

Example of Mapping XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<LabelFieldMap>
  <LabelField
Binding="/Shipment/SellerOrganization/CorporatePersonInfo/@AddressLine1"
LabelFieldName="FromAddressLine1" RepeatingElement="" />
  <LabelField
Binding="/Shipment/SellerOrganization/CorporatePersonInfo/@AddressLine2"
LabelFieldName="FromAddressLine2" RepeatingElement="" />
  <LabelField
Binding="concat(/Shipment/SellerOrganization/CorporatePersonInfo/@FirstName,
' ',/Shipment/SellerOrganization/CorporatePersonInfo/@LastName)"
LabelFieldName="FromName" RepeatingElement="" DataType="Text" />
  <LabelField
Binding="/Shipment/SellerOrganization/CorporatePersonInfo/@City"
LabelFieldName="FromCity" RepeatingElement="" DataType="Text" />
  <LabelField
Binding="/Shipment/SellerOrganization/CorporatePersonInfo/@State"
LabelFieldName="FromState" RepeatingElement="" DataType="Text" />
  <LabelField
Binding="/Shipment/SellerOrganization/CorporatePersonInfo/@Country"
LabelFieldName="FromCountry" RepeatingElement="" DataType="Text" />
  <LabelField
Binding="/Shipment/SellerOrganization/CorporatePersonInfo/@ZipCode"
LabelFieldName="FromZip" RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/@ShipmentNo" LabelFieldName="ShipmentNo"
RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/@ActualShipmentDate"
LabelFieldName="ShipmentDate" RepeatingElement="" DataType="Date" />
  <LabelField Binding="concat(/Shipment/ToAddress/@FirstName,
',/Shipment/ToAddress/@LastName)" LabelFieldName="ToName"
RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/ToAddress/@AddressLine1"
LabelFieldName="ToAddressLine1" RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/ToAddress/@AddressLine2"
LabelFieldName="ToAddressLine2" RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/ToAddress/@City" LabelFieldName="ToCity"
RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/ToAddress/@State"
LabelFieldName="ToState" RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/ToAddress/@ZipCode"
LabelFieldName="ToZip" RepeatingElement="" DataType="Text" />
  <LabelField Binding="/Shipment/ToAddress/@Country"
LabelFieldName="ToCountry" RepeatingElement="" DataType="Text" />
  <LabelField
Binding="concat(/Shipment/BillingInformation/AlternateParty/@FirstName,
',/Shipment/BillingInformation/AlternateParty/@LastName)"
LabelFieldName="BillToName" RepeatingElement="" DataType="Text" />
  <LabelField
Binding="/Shipment/BillingInformation/AlternateParty/@AddressLine1"
LabelFieldName="BillToAddressLine1" RepeatingElement="" DataType="Text" />
  <LabelField
```

```

Binding="/Shipment/BillingInformation/AlternateParty/@AddressLine2"
LabelFieldName="BillToAddressLine2" RepeatingElement="" DataType="Text"/>
  <LabelField Binding="/Shipment/BillingInformation/AlternateParty/@City"
LabelFieldName="BillToCity" RepeatingElement="" DataType="Text"/>
  <LabelField Binding="/Shipment/BillingInformation/AlternateParty/@State"
LabelFieldName="BillToState" RepeatingElement="" DataType="Text"/>
  <LabelField
Binding="/Shipment/BillingInformation/AlternateParty/@ZipCode"
LabelFieldName="BillToZip" RepeatingElement="" DataType="Text"/>
  <LabelField
Binding="/Shipment/BillingInformation/AlternateParty/@Country"
LabelFieldName="BillToCountry" RepeatingElement="" DataType="Text"/>
  <LabelField Binding="/Shipment/Carrier/@ScacDesc" LabelFieldName="SCAC"
RepeatingElement="" DataType="Text"/>
  <LabelField Binding="/Shipment/BillingInformation/@ShipmentChargeType"
LabelFieldName="FreightTerms" RepeatingElement="" DataType="Text"/>
  <LabelField Binding="concat(/Shipment/MarkForAddress/@FirstName,
',/Shipment/MarkForAddress/@LastName)" LabelFieldName="MarkFor"
RepeatingElement="" DataType="Text"/>
  <LabelField
Binding="/Shipment/Instructions/Instruction[@InstructionType='SHIP']/@Instru
ctionText" LabelFieldName="Special Instruction" RepeatingElement=""
DataType="Text"/>
  <LabelField
Binding="/Shipment/ShipmentLines/ShipmentLine/OrderLine/@CustomerPONo"
LabelFieldName="CustomerPONo" RepeatingElement="ShipmentLine"
DataType="Text"/>
  <LabelField Binding="/Shipment/ShipmentLines/ShipmentLine/@ItemID"
LabelFieldName="ItemId" RepeatingElement="" DataType="Text"/>
  <LabelField
Binding="/Shipment/ShipmentLines/ShipmentLine/OrderLine/Item/@CustomerItem"
LabelFieldName="CustItemId" RepeatingElement="ShipmentLine"
DataType="Text"/>
  <LabelField
Binding="/Shipment/ShipmentLines/ShipmentLine/OrderLine/Item/@ItemDesc"
LabelFieldName="ItemDesc" RepeatingElement="ShipmentLine" DataType="Text"/>
  <LabelField
Binding="/Shipment/ShipmentLines/ShipmentLine/@UnitOfMeasure"
LabelFieldName="UOM" RepeatingElement="ShipmentLine" DataType="Text"/>
  <LabelField Binding="/Shipment/ShipmentLines/ShipmentLine/@OrderedQty"
LabelFieldName="OrdQty" RepeatingElement="ShipmentLine" DataType="Text"/>
  <LabelField Binding="/Shipment/ShipmentLines/ShipmentLine/@Quantity"
LabelFieldName="Quantity" RepeatingElement="ShipmentLine" DataType="Text"/>
  <LabelField
Binding="/Shipment/ShipmentLines/ShipmentLine/@BackOrderedQty"
LabelFieldName="BOQty" RepeatingElement="ShipmentLine" DataType="Text"/>
  <LabelField Binding="" LabelFieldName="Line" RepeatingElement=""
Sequence="Y" DataType="Text"/>
  <RepeatingFields>
    <RepeatingField LabelFieldName="CustomerPONo" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="ItemId" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="CustItemId" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="ItemDesc" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="UOM" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="OrdQty" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="Quantity" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
    <RepeatingField LabelFieldName="BOQty" MaxFirstPage="12"
MaxLastPage="12" MaxMidPage="12"/>
  </RepeatingFields>

```

```

        <RepeatingField LabelFieldName="Line" MaxFirstPage="12"
            MaxLastPage="12" MaxMidPage="12"/>
    </RepeatingFields>
</LabelFieldMap>

```

Editing and Relocating the Map File Generated for a Label Procedure

1. Edit the map file (XML) generated for a label (LWL) to associate the XML data to the fields required on the label:

Note: XPath Functions can be used in the binding, provided the XPath Binding for a RepeatingField represents a Nodeset.

- **Sequence** - Sequence="Y" setting is to be used in instances where a labelfield represents a sequence of numbers. For example, serial numbers in a table.
- **DataType** - Set up the relevant DataType for the LabelField. Valid values are Text, Date, and DateTime.
- **Repeating Element** - Specify the RepeatingElement for the XPath Binding. If no Repeating Element is specified, the element containing the attribute is used as the RepeatingElement by default. In this example, the ShipmentLine is the RepeatingElement:

```

<LabelField Binding="/Shipment/ShipmentLines/ShipmentLine/OrderLine/
Item/@ItemDesc" LabelFieldName="ItemDesc"
RepeatingElement="ShipmentLine" DataType="Text"/>

```

2. Place the edited XML map file into the Sterling Selling and Fulfillment Foundation Runtime Template folder by pasting it into the Runtime > Template > Label > Extn directory.

Defining Custom Print Services

Services required for printing a Pack List are supplied by default within the Sterling Selling and Fulfillment Foundation framework. The examples provided in the following topic can be used as a reference point to create custom Prints.

Configuring a Print Pack List Service

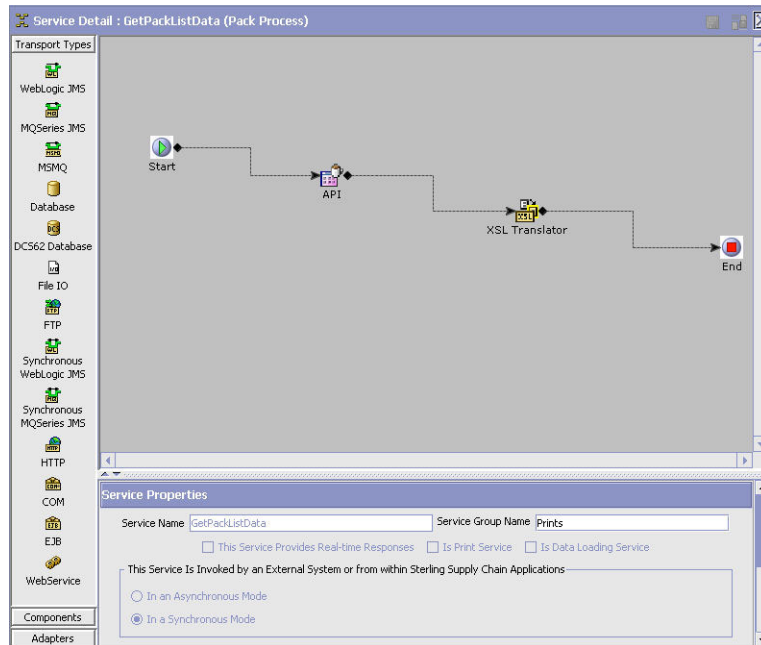
About this task

Prints are required to be configured as services to be invoked from an event or the console (UI).

To configure a Print Pack List service:

Procedure

1. From the Application Platform tree, choose Process Modeling > Container > Pack Process. The Pack Process window is displayed.
2. Choose Actions tab. From Pack Process Repository > Prints, choose PrintPackList.
3. The Service Detail: PrintPackList (Pack Process) window is displayed.



Results

For more information about configuring Service Details, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

The Input XML to the service definition is transformed into the input of the `PrintDocumentSet()` API using an XSL Translator.

For more information about the input to `PrintDocumentSet()` API and the description of the XML attributes, refer to the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Example of Typical XSL for `PrintDocumentSet()`

The following is an example of a typical XSL that generates the input to the `PrintDocumentSet()` API:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  version = "1.0">
  <xsl:output indent="yes"/>
  <xsl:template match="Print | Shipment">
  <PrintDocuments>
  <xsl:attribute name="PrintName">
  <xsl:text>packList</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="FlushToPrinter">
  <xsl:text>Y</xsl:text>
  </xsl:attribute>
  <PrintDocument>
  <xsl:attribute name="BeforeChildrenPrintDocumentId">
  <xsl:text>PACKLIST</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="DataElementPath">
  <xsl:text>xml:/Shipment</xsl:text>
  </xsl:attribute>
  <xsl:choose>
  <xsl:when test="name()='Print'">
  <xsl:copy-of select="PrinterPreference"/>
```

```

<xsl:copy-of select="LabelPreference"/>
</xsl:when>
<xsl:when test="name()='&quot;Shipment&quot;'>
<PrinterPreference>
<xsl:attribute name="PrinterId"/>
<xsl:attribute name="UsergroupId"/>
<xsl:attribute
  name="UserId"><xsl:text>xml:/Shipment/@Modifyuserid</xsl:text>
</xsl:attribute>
<xsl:attribute name="WorkStationId"/>
<xsl:attribute
  name="OrganizationCode"><xsl:text>xml:/Shipment/ShipNode/
  @NodeOrgCode</xsl:text></xsl:attribute>
</PrinterPreference>
<LabelPreference>
<xsl:attribute name="EnterpriseCode">
<xsl:text>xml:/Shipment/@EnterpriseCode</xsl:text>
</xsl:attribute>
<xsl:attribute name="BuyerOrganizationCode">
<xsl:text>xml:/Shipment/@BuyerOrganizationCode</xsl:text>
</xsl:attribute>
<xsl:attribute name="SellerOrganizationCode">
<xsl:text>xml:/Shipment/@SellerOrganizationCode</xsl:text>
</xsl:attribute>
</LabelPreference>
</xsl:when>
</xsl:choose>
<KeyAttributes>
<KeyAttribute
  name="Name"><xsl:text>ShipmentKey</xsl:text></xsl:attribute>
</KeyAttribute>
</KeyAttributes>
<InputData>
<xsl:attribute name="FlowName">
<xsl:text>GetPackListData</xsl:text>
</xsl:attribute>
<Shipment>
<xsl:choose>
<xsl:when test="name()='&quot;Print&quot;'>
<xsl:copy-of select="Shipment/@*" />
</xsl:when>
<xsl:when test="name()='&quot;Shipment&quot;'>
<xsl:copy-of select="@*" />
</xsl:when>
</xsl:choose>
</Shipment>
<Template>
<Api Name="getShipmentDetails">
<Template>
<Shipment>
<SellerOrganization>
<CorporatePersonInfo/>
</SellerOrganization>
<Carrier/>
<MarkForAddress/>
<BillingInformation>
<AlternateParty/>
</BillingInformation>
<Instructions>
<Instruction/>
</Instructions>
<FromAddress/>
<ToAddress/>
<ShipmentLines>
<ShipmentLine CountryOfOrigin="" FifoNo="" ItemDesc="" ItemID=""
  OrderHeaderKey="" OrderLineKey="" OrderNo="" OrderReleaseKey=""
  PrimeLineNo="" ProductClass="" Quantity="" ReleaseNo="" Segment=""

```

```

        SegmentType="" ShipmentKey="" ShipmentLineKey="" ShipmentLineNo=""
        SubLineNo="" UnitOfMeasure="" BackOrderedQty="" ShipmentSubLineNo="">
</Order/>
</OrderLine>
</Item/>
<OrderStatuses>
<OrderStatus OrderHeaderKey="" OrderLineKey="" OrderLineScheduleKey=""
    OrderReleaseKey="" OrderReleaseStatusKey="" PipelineKey=""
    ReceivingNode="" ShipNode="" Status="" StatusDate=""
    StatusDescription="" StatusQty="" StatusReason="" TotalQuantity="">
<OrderStatusTranQuantity StatusQty="" TotalQuantity=""
    TransactionalUOM="" />
<Details ExpectedDeliveryDate="" ExpectedShipmentDate="" ShipByDate=""
    TagNumber="">
</Details>
</OrderStatus>
</OrderStatuses>
</OrderLine>
</ShipmentLine>
</ShipmentLines>
<Containers>
<Container>
<ContainerDetails>
<ContainerDetail>
<ShipmentLine>
<OrderLine>
</Item/>
</OrderLine>
</ShipmentLine>
</ContainerDetail>
</ContainerDetails>
</Container>
</Containers>
<ShipNode>
<ShipNodePersonInfo/>
</ShipNode>
</Shipment>
</Template>
</Api>
</Template>
</InputData>
</PrintDocument>
</PrintDocuments>
</xsl:template>
</xsl:stylesheet>

```

Format for Input XML to XSL Translator

The Input XML to the above XSL translator should belong to either of the following formats:

```
<Shipment ShipmentKey=""/>
```

OR

```
<Print><Shipment ShipmentKey=""/><LabelPreference EnterpriseCode=""/><Printer
Preference UserId="" UsergroupId=""/></Print>
```

The former input XML is passed when the service is invoked from an event, while the latter is passed when the service is invoked from the console (UI).

XML Generated After XSL Translation

The following is an example of the XML generated after the XSL Translation using the above mentioned XSL:


```

<?xml version = "1.0" encoding = "UTF-8"?>
<PrintDocuments PrintName="packList" FlushToPrinter="Y">
<PrintDocument Localecode="xml:/Shipment/ShipNode/@Localecode">
<InputData APIName="getShipmentDetails">
<Shipment ShipmentKey="">
</Shipment>
<Template>
<Shipment>
<ShipNode>
<ShipNodePersonInfo/>
</ShipNode>
</Shipment>
</Template>
</InputData>
</PrintDocument>
<PrintDocument BeforeChildrenPrintDocumentId="PACKLIST"
  DataElementPath="xml:/Shipment">
<PrinterPreference PrinterId="" UserId="xml:/Shipment/@Modifyuserid"
  UsergroupId="" WorkStationId=""
  OrganizationCode="xml:/Shipment/ShipNode/@NodeOrgCode"/>
<LabelPreference EnterpriseCode="xml:/Shipment/@EnterpriseCode"
  BuyerOrganizationCode="xml:/Shipment/@BuyerOrganizationCode"
  SellerOrganizationCode="xml:/Shipment/@SellerOrganizationCode" />
<KeyAttributes>
<KeyAttribute Name="ShipmentKey"/>
</KeyAttributes>
<InputData FlowName="GetPackListData">
<Shipment ShipmentKey=""/>
<Template>
<Api Name="getShipmentDetails">
<Template>
<Shipment ShipmentKey="" ShipmentNo="" ActualShipmentDate=""
  ExpectedShipmentDate="">
<SellerOrganization OrganizationCode="">
<CorporatePersonInfo AddressLine1="" AddressLine2="" FirstName=""
  MiddleName="" LastName="" City="" State="" Country="" ZipCode="" />
</SellerOrganization>
<Carrier Scac="" ScacDesc=""/>
<MarkForAddress/>
<BillingInformation ShipmentChargeType=""/>
<Instructions>
<Instruction InstructionType="" InstructionText=""/>
</Instructions>
<ToAddress/>
<ShipmentLines>
<ShipmentLine ItemDesc="" ItemID="" OrderHeaderKey="" OrderLineKey=""
  OrderNo="" OrderReleaseKey="" PrimeLineNo="" Quantity="" ReleaseNo=""
  ShipmentKey="" ShipmentLineKey="" ShipmentLineNo="" SubLineNo=""
  UnitOfMeasure="" BackOrderedQty="" ShipmentSubLineNo="">
<Order OrderHeaderKey="" OrderNo="">
<PersonInfoBillTo AddressLine1="" AddressLine2="" FirstName="" MiddleName=""
  LastName="" City="" State="" Country="" ZipCode="" />
</Order>
<OrderLine CustomerPONO="" OrderLineKey="" OrderedQty=""
  OriginalOrderedQty="" Status="" StatusQuantity="" SubLineNo="" >
<Item CustomerItem=""/>
<OrderStatuses>
<OrderStatus OrderLineKey="" OrderReleaseStatusKey="" Status=""
  StatusQty="" TotalQuantity=""/>
</OrderStatuses>
</OrderLine>
</ShipmentLine>
</ShipmentLines>
<ShipNode NodeOrgCode=""/>
</Shipment>
</Template>
</Api>

```

```

</Template>
</InputData>
</PrintDocument>
</PrintDocuments>

```

Printing a Packing Slip with the GetPackListData Service

About this task

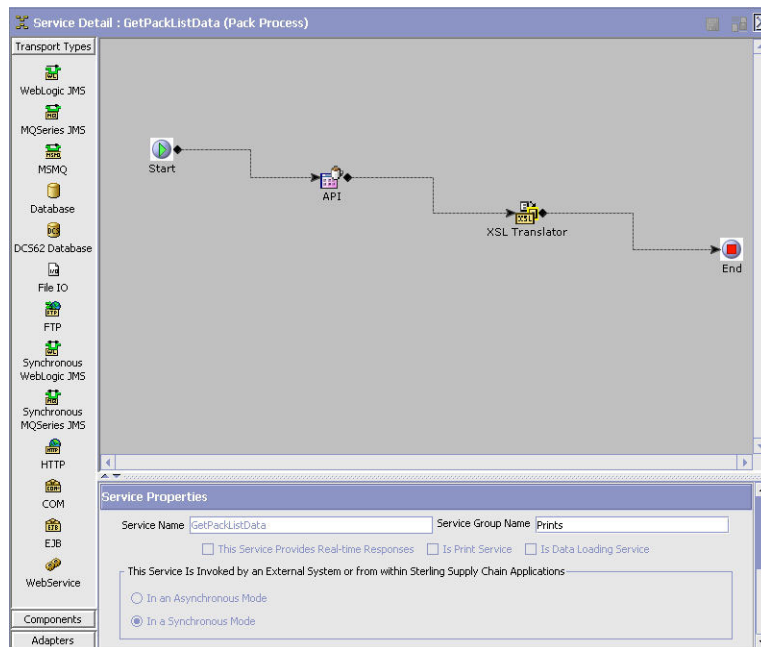
The XML shown in XML Generated After XSL Translation prints a Packing Slip (PACKLIST) as specified by the BeforeChildrenPrintDocumentId attribute in the PrintDocument node.

The data required to print the packlist is obtained by invoking the GetPackListData service as specified by the FlowName attribute in the InputData node.

To configure the GetPackListData service definition:

Procedure

1. From the Application Platform tree, choose Process Modeling > Container > Pack Process. The Pack Process window is displayed.
2. Choose Service Definitions Tab. From Pack Process Repository > Prints, choose GetPackListData.
3. The Service Details: GetPackListData (Pack Process) window is displayed.



Results

For more information about configuring Service Details, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

The GetPackListData service calls the GetShipmentDetails() API and the output is transformed using the XSL Translator.

The XSL translator (as reproduced below) calculates the backordered quantity for the shipment lines returned by the GetShipmentDetails() API:

```

<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" version =
"1.0">
  <xsl:output indent="yes"/>
  <xsl:template match="/Shipment">
    <Shipment>
      <xsl:choose>
        <xsl:when test="not(@ActualShipmentDate) or
(@ActualShipmentDate=&quot;&quot;)">
          <xsl:attribute name="ActualShipmentDate"><xsl:value-of
select="@ExpectedShipmentDate"/></xsl:attribute>
        </xsl:when>
        <xsl:otherwise>
          <xsl:attribute name="ActualShipmentDate"><xsl:value-of
select="@ActualShipmentDate"/></xsl:attribute>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:message>ActualShipmentDate<xsl:value-of
select="@ActualShipmentDate"/></xsl:message>
      <xsl:for-each select="@*">
        <xsl:if test="not(name()= &quot;ActualShipmentDate&quot;)">
          <xsl:attribute name="{name()}"><xsl:value-of select="."/>
        </xsl:if>
      </xsl:for-each>
      <xsl:copy-of select="SellerOrganization"/>
      <xsl:copy-of select="Carrier"/>
      <xsl:copy-of select="ShipNode"/>
      <xsl:copy-of select="ToAddress"/>
      <xsl:copy-of select="MarkForAddress"/>
      <xsl:copy-of select="BillingInformation"/>
      <xsl:copy-of select="Instructions"/>
      <xsl:copy-of select="Containers"/>
      <ShipmentLines>
        <xsl:for-each select="ShipmentLines/ShipmentLine[@Shipment
SubLineNo='0']">
          <ShipmentLine>
            <xsl:variable name="qty"
select="sum(OrderLine/OrderStatuses/OrderStatus[@OrderLineKey=current()
/@OrderLineKey and substring(@Status,1,4)='1300']/@StatusQty)"/>
            <xsl:attribute name="OrderedQty">
              <xsl:value-of
select="sum(OrderLine/OrderStatuses/OrderStatus[@OrderLineKey=current()
/@OrderLineKey and not(substring(@Status,1,4)='1400']/@StatusQty)"/>
            </xsl:attribute>
            <xsl:attribute name="BackOrderedQty">
              <xsl:value-of select="$qty"/>
            </xsl:attribute>
            <xsl:copy-of select="@*"/>
            <xsl:copy-of select="OrderLine"/>
          </ShipmentLine>
        </xsl:for-each>
      </ShipmentLines>
    </Shipment>
  </xsl:template>
</xsl:stylesheet>

```

Once a service has been created for a print, it should be associated to an appropriate event. For more information about Service Association, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Configuration Guide*.

Chapter 6. Integrating with Parcel Carrier Adapters

Overview of Parcel Carrier Adapter Integration

The Parcel Carrier Adapters (Carrier Adapter) manages all the carrier integration-related functions of Sterling Selling and Fulfillment Foundation. Sterling Selling and Fulfillment Foundation interfaces with the Carrier Adapter to use its carrier-integration functions.

The Carrier Adapter is regularly updated with the latest carrier data, such as rates and special services, and can act as a centralized carrier-integration database and business rules manager. The Carrier Adapter helps you to quickly meet the changing requirements initiated by both carriers and customers, in the most efficient way.

The Carrier Adapter has a data-driven design. The functionality is defined in terms of the relations between data elements stored in the database. Carriers having similar functionality can be incorporated into an installation with minimal engineering effort.

The Carrier Adapter is now integrated into Sterling Selling and Fulfillment Foundation. For more information about the Carrier Adapter and how to configure it, see the *Sterling Selling and Fulfillment Foundation: Parcel Carrier Adapter* .

APIs Invoked During Parcel Carrier Adapter Integration

These APIs are invoked during the Sterling Warehouse Management System integration with the Carrier Adapter.

APIs Invoked During the Carrier Adapter Integration with UPSN

- openManifest API
- shipCarton API
- deleteCarton API
- closeManifest API

APIs Invoked During the Carrier Adapter Integration with FedEx

- openManifest API
- shipCarton API
- deleteCarton API
- closeManifest API

APIs Used for Carrier Adapter Integration

The Sterling Warehouse Management System integrates with the Carrier Adapter using the following APIs:

- **openManifest API:** The openManifest API is used to open a manifest for a carrier server. This API calls the openManifest API in the Carrier Adapter. For field level mapping details between these APIs, see "Field-Level Mapping Between the openManifest API on the Sterling Warehouse Management System and the openManifest API on the Carrier Adapter".

- **addContainerToManifest API:** The addContainerToManifest API is used to add a container to a manifest. This API calls the shipCarton API in the Carrier Adapter. For field level mapping details between these APIs, see "Mappings Between the addContainerToManifest API on the Sterling Warehouse Management System and the shipCarton API on the Carrier Adapter".
- **removeContainerFromManifest API:** The removeContainerFromManifest API is used to delete a carton from a manifest. This API calls the deleteCarton API on the Carrier Adapter. For field level mapping details between these APIs, see "Mapping Between the removeContainerFromManifest API on the Sterling Warehouse Management System and the deleteCarton API on the Carrier Adapter".
- **closeManifest API:** The closeManifest API is used to close a manifest. This API calls the closeManifest API on the Carrier Adapter. For field level mapping details between these APIs, see "Mapping Between the closeManifest API on the Sterling Warehouse Management System and the closeManifest API on the Carrier Adapter".

Note: For the FedEx carrier, the Carrier Adapter supports label prints when a container is added to a manifest if the FedEx Printer is configured on the FedEx Carrier Server. For the UPSN carrier, the Carrier Adapter supports label prints when a container is added to a manifest or a manifest is closed. For more information about Label Prints, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System User Guide*.

Field-Level Mapping Between the openManifest API on the Sterling Warehouse Management System and the openManifest API on the Carrier Adapter

This is the input XML for the field-level mapping between the openManifest API on the Sterling Warehouse Management System and the openManifest API on the Carrier Adapter:

Field Name	Comments	Platform
Carrier	Required	YFS_Manifest.SCAC
ManifestNumber	Required	YFS_MANIFEST.manifest_no (as entered by the user. If not entered, posted with one upsequence number generated)
PickupSummaryNumber	Required for UPSN	YFS_MANIFEST.pickup_summary_no (as entered by the user)
ShipperAccountNumber	Required	YFS_MANIFEST.shipper_account_no (as entered by the user)
PickupDate	Required	YFS_MANIFEST.manifest_date (as entered by the user)

No output XML is generated for the openManifest API. A confirmation message is displayed on success, while an error message is displayed in the event of a failure.

Mappings Between the addContainerToManifest API on the Sterling Warehouse Management System and the shipCarton API on the Carrier Adapter

This is the input XML for the field-level mapping between the addContainerToManifest API on the Sterling Warehouse Management System and the shipCarton API on the Carrier Adapter:

Field Name	Comments	Platform
UPSPLD		
Carrier	Required	YFS_SHIPMENT.scac
PackageLevelDetail	The Package Level Detail Record (0100) - is written for every package shipped. This is a mandatory record for both domestic and international shipments.	
ManifestNumber	Required	YFS_MANIFEST.manifest_no (open manifest as obtained by packShipment API for a given shipnode and carrier)
ShipId	Required	YFS_SHIPMENT_CONTAINER.container_no.
PickupDate	Required	YFS_MANIFEST.manifest_date
ShipperAccountNumber	Required	YFS_MANIFEST.shipper_account_no
BookNumber	Required	YFS_MANIFEST.pickup_summary_no (substring 0-7)
PageNumber	Required	YFS_MANIFEST.pickup_summary_no (substring 8-10)
ShipmentNumber	Required	YFS_SHIPMENT.shipment_no
PackageTrackingNumber	Required	<spaces>
SPFVersion	Required	Default 0505
Acctnumber	Conditional	Computed based on YFS_FREIGHT_TERMS.charges_paid_by. It can be YFS_SHIPMENT.Custcarrier_Account_No/YFS_SCAC_ Ex.account1.
CompanyName	Required	YFS_PERSON_INFO.company corresponding to YFS_SHIPMENT.to_address_key.
ConsigneeAttn	Conditional	YFS_PERSON_INFO.first_name + YFS_PERSON_INFO.middle_name + YFS_PERSON_INFO.last_name corresponding to YFS_SHIPMENT.to_address_key.
CAddr1	Required	YFS_PERSON_INFO.address_line1 corresponding to YFS_SHIPMENT.to_address_key.

Field Name	Comments	Platform
CAddr2	Optional	YFS_PERSON_INFO.address_line2 corresponding to YFS_SHIPMENT.to_address_key.
CAddr3	Optional	YFS_PERSON_INFO.address_line3 corresponding to YFS_SHIPMENT.to_address_key.
CCity	Required	YFS_PERSON_INFO.city corresponding to YFS_SHIPMENT.to_address_key.
CStateProv	Conditional	YFS_PERSON_INFO.state corresponding to YFS_SHIPMENT.to_address_key.
CPostalCode	Conditional	YFS_PERSON_INFO.zip_code corresponding to YFS_SHIPMENT.to_address_key.
CPhone	Conditional	YFS_PERSON_INFO.day_phone corresponding to YFS_SHIPMENT.to_address_key.
ShipmentChgType	Required	Computed based on YFS_FREIGHT_TERMS.charges_paid_by and corresponding YFS_SCAC_Ex entry. Possible values are COL,TPB,PRE.
CWTInd	Conditional	Set to '0' (zero) to indicate Not HunderedWeight.
ServiceType	Required	YFS_SCAC_AND_SERVICE.electronic_code corresponding to YFS_SHIPMENT.scac and YFS_SHIPMENT.carrier_service_code.
Packagetype	Required	02" to indicate Package.
DeliveryZone	Optional	<spaces>
Actualweight	Required	YFS_SHIPMENT_CONTAINER.container_gross_weight after applying the carrier locale weight UOM.
PkgpublishedDimWt	Required	Computed
UOMWeight	Optional	Weight UOM of the Ship Node
UOMDim	UOM Dim	Dimension UOM of the Ship Node
CODAmount	Required	0
CODFundsInd	Conditional	<spaces>
Currencycode	Required	YFS_SHIPMENT.currency.
CallTag_ARSIInd	Required	0 - to indicate no call tag.
Calltag_ARSSchedulePickDate	Optional	<spaces>
MerchandiseDescription	Conditional	<spaces>

Field Name	Comments	Platform
SatDeliveryInd	Required	0" for not opting for this service.
SaturdayPickupInd	Required	0" for not opting for this service.
OversizePackageInd	Required	YFS_SHIPMENT_CONTAINER.oversized_flag is Y, then indicator is passed as 1, or else 0.
DeclaredValueInsurance	Required	YFS_SHIPMENT_CONTAINER.declared_value
ResInd	Required	YFS_PERSON_INFO.company corresponding to YFS_SHIPMENT.to_address_key is nonblanks, it is assumed to be 0 to indicate commercial or else 1 for residential.
DCISType	Conditional	<spaces>
CustomerRefNumberType1	Optional	<spaces>
CustomerRefNumber1	Optional	<spaces>
CustomerRefNumberType2	Optional	<spaces>
CustomerRefNumber2	Optional	<spaces>
ShipmentReferenceNoType1	Optional	<spaces>
ShipmentReferenceNo1	Optional	<spaces>
ShipmentReferenceNoType2	Optional	<spaces>
ShipmentReferenceNo2	Optional	<spaces>
CODControlNumber	Optional	<spaces>
CallTag_ARSNumber	Optional	<spaces>
CODInd	Required	<spaces>
CODCurrencycode	Conditional	<spaces>
IncrementalPldInd	Required	<spaces>
DocInd	Required	Default to '3' to indicate non document/package.
ShipperEIN	Optional	<spaces>
ShipperCountry	Required	YFS_PERSON_INFO.country corresponding to YFS_SHIPMENT.shipnode_key's YFS_SHIP_NODE.shipnode_address_key.
SenderName	Optional	<spaces>
ConsigneeTagID	Optional	<spaces>
ConsigneeCountry	Required	YFS_PERSON_INFO.country corresponding to YFS_SHIPMENT.to_address_key.
CalculatedRatesInd	Required	<spaces>
SourceTypeCode	Required	Default to '20' to indicate host access.
AccessorialRecord	AccessorialRecord (0200) is valid for both domestic and international shipments. This record is written only when UPS special services are used.	

Field Name	Comments	Platform
ShipperCreditCardNo	Required	<spaces>
ShipperCreditCardExpDate	Required	<spaces>
AdditionalHandlingInd	Required	Default to '0'.
ExtendedDestInd	Required	<spaces>
HazMat	Required	YFS_SHIPMENT.hazardous material is Y, then indicator is 1, else 0.
HoldForPickupInd	Required	Default to '0' (do not hold for pickup).
ModifyInd	Required	Default to '0'.
OCAIndicator	Required	Default to '0'.
VoidInd	Required	0
PackageLength	Required	YFS_SHIPMENT_CONTAINER.container_length
PackageWidth	Required	YFS_SHIPMENT_CONTAINER.container_width
PackageHeight	Required	YFS_SHIPMENT_CONTAINER.container_height
SpecialInstructions	Optional	<spaces>
VerbalConfirmationName	Conditional	YFS_PERSON_INFO.first_name + YFS_PERSON_INFO.middle_name + YFS_PERSON_INFO.last_name corresponding to YFS_SHIPMENT.to_address_key.
VerbalConfirmationPhone	Conditional	YFS_PERSON_INFO.day_phone corresponding to YFS_SHIPMENT.to_address_key.
EarliestDeliveryTime	Optional	<spaces>
ShipmentCreditCardNumber	Conditional	<spaces>
ShipmentCreditCardExpDate	Conditional	<spaces>
ConsigneeNumber	Optional	<spaces>
ConsigneeCreditCardNo	Required	<spaces>
ConsigneeCreditCardExpDate	Required	<spaces>
DCISNumber	Optional	<spaces>
ConsigneeFaxDestinationInd	Optional	<spaces>
ConsigneeFax	Optional	<spaces>
ExperssCODTrackingNumber	Required	<spaces>
CustomerReferenceNumberType3	Optional	<spaces>
CustomerReferenceNumber3	Optional	<spaces>
CustomerReferenceNumberType4	Optional	<spaces>
CustomerReferenceNumber4	Optional	<spaces>
CustomerReferenceNumberType5	Optional	<spaces>
CustomerReferenceNumber5	Optional	<spaces>
PackageTrackingNumber	Required	YFS_Shipment_Container.Tracking_No

Field Name	Comments	Platform
AlternatePartyRecord	AlternateParty Record (0300) is valid for both domestic and international shipments. For domestic, this record is written only when freight term is 'Third Party Billing'. For International shipments, this record is written for Importer and Exporter Address.	
AlternatePartyType	Required	For domestic shipments: This field is set to '03'/'04'. For international shipments: This field is set to '02' always.
ID_AcctNumber	Conditional	YFS_SCAC_EX.account1
PODReplyType	Conditional	<spaces>
APCompanyName	Required	YFS_PERSON_INFO.company corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.
APAttention	Conditional	YFS_PERSON_INFO.first_name + YFS_PERSON_INFO.last_name corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.
APAddr1	Required	YFS_PERSON_INFO.address_line1 corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.
APAddr2	Optional	YFS_PERSON_INFO.address_line2 corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.
APAddr3	Optional	YFS_PERSON_INFO.address_line3 corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.
APCity	Required	YFS_PERSON_INFO.city corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.
APStateProv	Conditional	YFS_PERSON_INFO.state corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key. Note: This field value can only contain a maximum of 5 characters.
APPostalCode	Conditional	YFS_PERSON_INFO.zip_code corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key.

Field Name	Comments	Platform
APcountry	Required	YFS_PERSON_INFO.country corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key. If International it is hardcoded to 'US'.
Filler1	Required	
APPhone	Conditional	YFS_PERSON_INFO.day_phone_no corresponding to YFS_SHIPMENT.enterprise_code's billing_address_key
APFaxDestInd	Conditional	<spaces>
APFax	Optional	<spaces>
LangCode	Optional	<spaces>
CreditCardNo	Required	<spaces>
CreditCardExpDate	Required	<spaces>
TaxId	Optional	<spaces>
AddrType	Required	<spaces>
PackageTrackingNumber	Required	YFS_Shipment_Container.Tracking_No
AdvisoryInformationRecord	AdvisoryInformationRecord (0400) is required for E-mail or Fax Shipment Notification.	
AdvisoryInfoLevel	Required	Default to 'P'.
SNFaxDestInd1	Conditional	If YFS_PERSON_INFO.day_fax_no != "" set this field to 0. US, PR, CA, and VI Fax/Phone only 1 Fax/Phone to all other countries.
SNFaxNumber1	Conditional	YFS_PERSON_INFO.day_fax_no corresponding to YFS_SHIPMENT.to_address_key.
SNLangCode	Optional	<spaces>
SNCompName1	Optional	YFS_PERSON_INFO.company corresponding to YFS_SHIPMENT.to_address_key.
SNAttnName1	Conditional	YFS_PERSON_INFO.first_name + YFS_PERSON_INFO.middle_name + YFS_PERSON_INFO.last_name corresponding to YFS_SHIPMENT.to_address_key.
SNContactPhone1	Conditional	YFS_PERSON_INFO.day_phone corresponding to YFS_SHIPMENT.to_address_key.
SNFaxDestInd2	Conditional	<spaces>
SNFaxNumber2	Conditional	<spaces>
SNLangCode2	Optional	<spaces>
SNCompanyName2	Optional	<spaces>

Field Name	Comments	Platform
SNAttnName2	Conditional	<spaces>
SNContactPhone2	Conditional	<spaces>
AltrofileAccessNumber	Required	<spaces>
SNTypeDestination1	Required	<spaces>
SNEmailAddrDest1	Conditional	YFS_PERSON_INFO.email_id corresponding to YFS_SHIPMENT.to_address_key.
SNTypeDestination2	Required	Set to '0'
SNEmailAddrDest2	Conditional	<spaces>
SNMemo	Optional	<spaces>
PackageTrackingNumber	Required	YFS_Shipment_Container.Tracking_No
InternationalRecord	InternationalRecord (0500) is required if Importer, Exporter, Shipper To Consignee, or Commodity information is provided and whenever shipper and consignee countries are not the same. This record is written once for one shipment. If a shipper has 3 packages, only one 0500 record is written, whereas three 0100 records are written.	
RecordType	Required	0500
InvoiceDate	Optional	YFS_MANIFEST.manifest_date (manifest no from YFS_SHIPMENT).
WaybillPrintInd	Conditional	0
InvoiceLineTotals	Required	YFS_CONTAINER_DETAILS.quantity * YFS_ORDER_LINE * unit_price (for all lines in the container).
InvoiceCurrencyCode	Conditional	YFS_SHIPMENT.currency
ShipmentInsuranceDeclaredValue	Required	YFS_MANIFEST.manifest_date (manifest no from YFS_SHIPMENT).
ConsolidatedClearQty	Required	0
UltimateDestCountry	Conditional	YFS_PERSON_INFO.country corresponding to YFS_SHIPMENT.to_address_key.
Filler		<spaces>
SEDCCode	Optional	<spaces>
ShipmentSEDCASNum	Optional	<spaces>
InvoiceNumber	Optional	YFS_SHIPMENT.shipment_no
PONumber	Optional	<spaces>
DescriptionOfGoods	Required	YFS_ITEM.nmfc_code. Item_Id taken from CONTAINER_DETAILS.item_id with YFS_SHIPMENT_CONTAINER.container_no (leadpackage) as criteria.
SpecialInstructions	Optional	<spaces>

Field Name	Comments	Platform
PartiesToTrans	Conditional	<spaces>
TermsOfShipment	Optional	<spaces>
PaymentTerms	Optional	<spaces>
Filler		<spaces>
FreightCharges	Required	0
InsuranceCharges<	Required	0
DiscountRebate	Required	0
OtherCharges	Required	0
WaybillNumber/BrokerageID	Conditional	YFS_SHIPMENT.shipment_no
COCode	Optional	<spaces>
OtherDocCode	Optional	<spaces>
ReasonForExport	Optional	<spaces>
InvoiceSubTotal	Required	<spaces>
TotalInvoiceAmount	Required	<spaces>
BrokerCode	Optional	<spaces>
DestinationControl	Conditional	<spaces>
ShipmentCommodityOrigin	Conditional	<spaces>
Filler3	Required	
PackageTrackingNumber	Required	<spaces>
CommodityRecord	CommodityRecord (0600) contains commodity information that is used for rating and customs clearance purposes. It is required if the shipment travels within the European Union and contains "Goods Not in Free Circulation". One 0600 record is written for each line in the shipper. If a shipper on the Sterling Warehouse Management System has 4 records in the YFS_SHIPMENT_DTL table, four 0600 records are written.	
RecordType	Required	0600
InvoiceLineNumber	Required	YFS_SHIPMENT_LINE.prime_line_no for the corresponding YFS_CONTAINER_DETAILS record.
CommodityCode	Optional	YFS_ITEM.harmonized_code of YFS_CONTAINER_DETAILS.item_id (catalog org and uom).
PartNumber	Optional	YFS_ITEM.item_id of YFS_CONTAINER_DETAILS.item_id (catalog org and uom).
LineOriginCountry	Required	YFS_ITEM.country_of_origin of YFS_CONTAINER_DETAILS.item_id (catalog org and uom).
LineCurrencyCode	Optional	YFS_SHIPMENT.currency

Field Name	Comments	Platform
ECCN	Optional	YFS_ITEM.eccn_no of YFS_CONTAINER_DETAILS.item_id (catalog org and uom).
LineUnitAmtPrice	Required	YFS_ORDER_LINE.line_price of YFS_CONTAINER_DETAILS.order_line_key * YFS_CONTAINER_DETAILS.quantity. If its shipment container, we compute by getting item object from shipment and shipment container.
LineQuantity	Required	sum(YFS_CONTAINER_DETAIL.quantity) for every unique item.
LineQtyUOM	Required	YFS_CONTAINER_DETAILS.uom
LineLicenseInfo	Conditional	YFS_SHIPMENT_CONTAINER.export_license_no
LineLicenseExpDate	Conditional	YFS_SHIPMENT_CONTAINER.export_license_exp_date
LineMerchDesc1	Required	YFS_ITEM.item_desc of YFS_CONTAINER_DETAILS.item_id (catalog org and uom).
LineMerchDesc2	Optional	<spaces>
LineMerchDesc3	Optional	<spaces>
CertOfOriginNo	Optional	YFS_SHIPMENT.shipment_no
CertOfOriginCode	Conditional	<spaces>
AgreementType	Optional	<spaces>
CommodityRemarks	Optional	<spaces>
QuantityScheduledUnits	Conditional	YFS_CONTAINER_DETAIL.quantity
Marks&Numbers	Optional	<spaces>
CommodityWeight	Required	YFS_ORDER_LINE.item_weight of YFS_CONTAINER_DETAILS.order_line_key * YFS_CONTAINER_DETAILS.quantity. If its shipment container, we compute by getting the item object from shipment and shipment container.
NumberOfPackagesPerCmmdty	Conditional	<spaces>
SEDLineAmt	Required	YFS_ORDER_LINE.line_price of YFS_CONTAINER_DETAILS.order_line_key * YFS_CONTAINER_DETAILS.quantity. If its shipment container, we compute by getting the item object from shipment and shipment container.
COType	Required	Defaulted to 0.

Field Name	Comments	Platform
SEDInd	Required	Defaulted to 0.
LineExtendedAmt	Required	YFS_ORDER_LINE.line_price of YFS_CONTAINER_DETAILS.order_line_key * YFS_CONTAINER_DETAILS.quantity. If its shipment conatiner, we compute by getting the item object from shipment and shipment container.
Filler		<spaces>
PackageTrackingNumber	Required	YFS_Shipment_Container.Tracking_No
AdditionalCommentsRecord	AdditionalCommentsRecord (0700) contains additional statements and information for an international shipment.	
RecordType	Required	0700
DeclarationStatement	Optional	<spaces>
AdditionalComments	Optional	<spaces>
Filler1		<spaces>
PackageTrackingNumber	Required	<spaces>
SpecialServicesRecord	SpecialServicesRecord contains SpecialService child elements for each of the special service the shipment/order have.	
Service	Optional	YFS_SPECIAL_SERVICE_REF.service_code
ExtraFieldsRecord	ExtraFieldsRecord contains statements and information extra fields.	
LableFormatValue	Optional	<spaces>
ReferenceNotes	Optional	YFS_SHIPMENT.shipment_no+ YFS_SHIPMENT_CONTAINER. container_Scm.
SunDeliveryInd	Optional	<spaces>
ThermalLabelPrinterID	Optional	Determined by calling getPrinterId.

Output XML for a Field-Level Mapping

This is the output XML for the field-level mapping between the addContainerToManifest API on the Sterling Warehouse Management System and the shipCarton API on the Carrier Adapter:

Field Name	Platform
TotalErrors	The total number of errors returned by the Carrier Server
ErrorCode	The error code returned by the Carrier Server
ErrorDescription	The description of the error code returned by the Carrier Server.

Field Name	Platform
CODReturnTrackingNo	YFS_SHIPMENT_CONTAINER.COD_Return_tracking_No
TrackingNumber	YFS_SHIPMENT_CONTAINER.tracking_no
TotalSurchargeAmt	YFS_SHIPMENT_CONTAINER.special_services_surcharge
NetCharge	YFS_SHIPMENT_CONTAINER.actual_freight_charge
BilledWeight	YFS_SHIPMENT_CONTAINER.applied_weight
PrintBuffer	The print buffer returned by the Carrier Server.
DeliveryDay	YFS_SHIPMENT_CONTAINER.delivery_day
UPS_Routing_Code	YFS_SHIPMENT_CONTAINER.UPS_Routing_Code

Mapping Between the removeContainerFromManifest API on the Sterling Warehouse Management System and the deleteCarton API on the Carrier Adapter

This is the mapping between the removeContainerFromManifest API on the Sterling Warehouse Management System and the deleteCarton API on the Carrier Adapter:

Field Name	Comments	Platform
Carrier	Required	YFS_SHIPMENT.scac
MeterNo	Required only for FedEx	YFS_SCACEx.portal_account_2
TrackingNumber	Required	YFS_SHIPMENT_CONTAINER.tracking_no of the package that is being unpacked or removed from the manifest.

No output XML is generated for the removeContainerFromManifest API. A confirmation message is displayed on success, while an error message is displayed in the event of a failure.

Mapping Between the closeManifest API on the Sterling Warehouse Management System and the closeManifest API on the Carrier Adapter

This is the input XML for the field-level mapping between the closeManifest API on the Sterling Warehouse Management System and the closeManifest API on the Carrier Adapter:

Field Name	Comments	Platform
Carrier	Required	YFS_SHIPMENT.scac
ManifestNumber	Required	YFS_MANIFEST.manifest_no (as generated on the platform for the ship node and carrier combination)
PickupSummaryNumber	Required for UPSN	YFS_MANIFEST.pickup_summary_no (as keyed in from the user)

Field Name	Comments	Platform
ShipperAccountNumber	Required	YFS_MANIFEST.shipper_account_no

No output XML is generated for the closeManifest API. A confirmation message is displayed on success, while an error message is displayed in the event of a failure.

Integration Dependencies

Sterling Warehouse Management System integration with the Carrier Adapter is dependent on the following:

- Carrier Adapter APIs are called only if SCAC Integration is required for the Shipment. This is set up at Node/SCAC level.

Chapter 7. Integrating with Material Handling Equipment

Overview of Material Handling Equipment Integration

The Sterling Warehouse Management System can integrate with various material handling equipment (MHE).

The automation enabled through the integration enables increased efficiency in various processes of a warehouse, like Receiving, Picking, Packing, Putaway or Replenishment, Outbound QC, VAS, Manifesting, Weighing, Item Measurements, and Trailer Loading.

The material handling equipment that the Sterling Warehouse Management System can integrate with includes the following:

- Pick-to-Light
- Put-to-Light
- Carousels or Automated Storage & Retrieval Systems (ASRS)
- Automatic Guided Vehicles (AGV)
- Inbound Sorters
- Pack Sorters
- Shipping Sorters
- Cube-a-Scans
- Weighing Scales

Integrating with Pick-to-Light Systems

The Sterling Warehouse Management System integrates with the pick-to-light systems after the Sterling Warehouse Management System allocates and creates pick/move tasks.

1. For tasks that are in the pick-to-light zone, details regarding shipment/batch/carton (reference tag) level that indicate item and quantity to pick are sent to the system.

APIs involved are:

- createTask()
- changeTask()
- createBatch()
- getTaskList()
- cancelTask()

The following event is raised by the createTask() API:

- CREATE_TASK.TASK_CREATED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

The following event is raised by the createBatch() API:

- CREATE_BATCH.BATCH_CREATED

The following event is raised by the cancelTask() API:

- CANCEL_TASK.TASK_CANCELED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. References are scanned in the pick-to-light system and appropriate slots are lit indicating quantity to pick.
2. Upon pick completion, status information is sent from the pick-to-light system to the Sterling Warehouse Management System. All serial/tag number level information required for pick completion is also passed back to the Sterling Warehouse Management System.

The APIs involved are:

- registerTaskCompletion()
- registerBatchCompletion()
- changeTask()

The following events are raised by the registerTaskCompletion() and registerBatchCompletion() APIs:

- COMPLETE_TASK.TASK_COMPLETED
- COMPLETE_BATCH.BATCH_COMPLETED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Integrating with Put-to-Light Systems

The Sterling Warehouse Management System integrates with the put-to-light systems after the Sterling Warehouse Management System allocates and creates pick/move tasks after wave release.

1. For tasks that are in the put-to-light zone, details regarding shipment/order level that indicate item and quantity to pick are sent to the system. The Sterling Warehouse Management System is configured to create the required number of shipments in a wave, to match the number of slots.

The APIs involved are:

- getShipmentDetails()
- createTask()
- changeTask()
- createBatch()
- getTaskList()
- cancelTask()

The following event is raised by the createTask() API:

- CREATE_TASK.TASK_CREATED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

The following event is raised by the createBatch() API:

- CREATE_BATCH.BATCH_CREATED

The following event is raised by the cancelTask() API:

- CANCEL_TASK.TASK_CANCELED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. Item Ids are scanned in the put-to-light system, and appropriate slots are lit indicating quantity to be placed.
2. Container numbers are associated to each slot, and the container is closed. This information is sent back to the Sterling Warehouse Management System.

The APIs involved are:

- registerTaskCompletion()
- registerBatchCompletion()
- addToContainer()
- changeTask()

The following events are raised by the registerTaskCompletion() and registerBatchCompletion() APIs:

- COMPLETE_TASK.TASK_COMPLETED
- COMPLETE_BATCH.BATCH_COMPLETED

The following events are raised by the addToContainer() API:

- CREATE_CONTAINER.ON_SUCCESS
- ADD_TO_CONTAINER.ON_SUCCESS
- ADD_TO_CONTAINER.ON_CONTAINER_PACK_COMPLETE
- ADD_TO_CONTAINER.ON_CONTAINER_PACK_PROCESS_COMPLETE
- ADD_TO_CONTAINER.ON_SHIPMENT_PACK_COMPLETE
- ADD_TO_CONTAINER.ON_SHIPMENT_PACK_PROCESS_COMPLETE

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. Quantities at the shipment level (each slot) are taken to appropriate packing locations to complete packing steps.

Integrating with Carousel or Automated Storage and Retrieval Systems

Integration When a Product is Being Put Away

When a product is being put away, the Sterling Warehouse Management System integrates with Carousels or Automated Storage and Retrieval Systems (ASRS) as follows:

1. The first step task brings the product to the drop-off location attached to the carousel/ASRS location. Upon completion of this task secondary step tasks are created. These secondary tasks based on task type and zone are sent to the carousel system.

The APIs involved are:

- createTask()
- createBatch()

The following event is raised by the createTask() API:

- CREATE_TASK.TASK_CREATED

The following event is raised by the createBatch() API:

- CREATE_BATCH.BATCH_CREATED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. User scans item for putaway into carousel system, which retrieves appropriate location/bin to the user station. Product is placed in the bin.
2. Upon the location/bin being placed in appropriate slot, the task completion information is sent to WMS. All serial/tag number level information required for pack completion is also passed back to WMS

The APIs involved are:

- registerTaskCompletion()
- registerBatchCompletion()
- changeTask()

The following events are raised by the registerTaskCompletion() and registerBatchCompletion() APIs:

- COMPLETE_TASK.TASK_COMPLETED
- COMPLETE_BATCH.BATCH_COMPLETED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Integration When a Product is Being Retrieved

When a product is being retrieved, the Sterling Warehouse Management System integrates with Carousels or Automated Storage and Retrieval Systems (ASRS) as follows:

1. Tasks created to retrieve product from the carousel/ASRS are sent from the Sterling Warehouse Management System.

The APIs involved are:

- createTask()
- createBatch()

The following event is raised by the createTask() API:

- CREATE_TASK.TASK_CREATED

The following event is raised by the createBatch() API:

- CREATE_BATCH.BATCH_CREATED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. User initiates retrieval on carousel system and selects task for retrieval. On retrieval, system sends completion of task from bin/location to drop-off location at user station.

The APIs involved are:

- registerTaskCompletion()
- registerBatchCompletion()
- changeTask()

The following events are raised by the registerTaskCompletion() and registerBatchCompletion() APIs:

- COMPLETE_TASK.TASK_COMPLETED
- COMPLETE_BATCH.BATCH_COMPLETED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. Secondary step tasks are automatically created by the Sterling Warehouse Management System to putaway quantity to final destination location.

Integration When a Product is Being Counted

When a product is being counted, the Sterling Warehouse Management System integrates with Carousels or Automated Storage and Retrieval Systems (ASRS) as follows:

- User on the Sterling Warehouse Management System is given location to count.
- This is entered on carousel system for location retrieval.
- Count is completed on the Sterling Warehouse Management System.

Integrating with Automatic Guided Vehicles

The Sterling Warehouse Management System integrates with Automatic Guided Vehicles (AGV) to complete putaway or pick. These interfaces are task-based integrations.

The APIs involved are:

- createTask()
- changeTask()
- createBatch()
- getTaskList()
- cancelTask()

The following event is raised by the createTask() API:

- CREATE_TASK.TASK_CREATED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

The following event is raised by the createBatch() API:

- CREATE_BATCH.BATCH_CREATED

The following event is raised by the cancelTask() API:

- CANCEL_TASK.TASK_CANCELED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Upon completion of task the confirmation is sent back to the Sterling Warehouse Management System.

The APIs involved are:

- registerTaskCompletion()
- registerBatchCompletion()
- changeTask()

The following events are raised by the registerTaskCompletion() and registerBatchCompletion() APIs:

- COMPLETE_TASK.TASK_COMPLETED
- COMPLETE_BATCH.BATCH_COMPLETED

The following events are raised by the changeTask() API:

- CHANGE_TASK.TASK_CHANGED
- CHANGE_TASK.TASK_PUT_ON_HOLD
- CHANGE_TASK.TASK_RELEASED_FROM_HOLD

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Integrating with Inbound Sorters

Inbound Sorters are typically used when expected LPN information is available on WMS.

The Sterling Warehouse Management System integrates with the inbound sorters as follows:

1. A shipment/ASN captures expected quantities. User indicates start of receipt of the ASN when container/truck pulls into the dock door. Information for the ASN is sent to sorter system along with lane sorting information, if applicable.

The APIs involved are:

- startReceipt()
- getShipmentDetails()
- getActivityDemand()

The following event is raised by the startReceipt() API:

- START_RECEIPT.ON_START_RECEIPT

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. LPNs are sorted to respective destination zones based on QC profiling and product characteristics.
2. The Sterling Warehouse Management System is notified when LPN reaches destination.

The API involved is:

- receiveOrder()

The following events are raised by the receiveOrder() API:

- RECEIVE_RECEIPT.ON_SUCCESS
- RECEIVE_RECEIPT.ON_SKU_RECEIPT
- RECEIVE_RECEIPT.ON_CASE_RECEIPT
- RECEIVE_RECEIPT.ON_PALLET_RECEIPT
- RECEIVE_ORDER.INVENTORY_COST_CHANGE
- RECEIVE_ORDER.INVENTORY_COST_WRITEOFF
- RECEIVE_ORDER.INVENTORY_VALUE_CHANGE

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. Putaway task is automatically generated on the Sterling Warehouse Management System.

Integrating with Pack Sorters

Pack sorters are used when loose items are picked and need to be sent to pack stations.

The Sterling Warehouse Management System integrates with pack sorters as follows:

1. A tag indicating the shipment is associated with the pick before placing on the conveyor system.

2. Data is published to sorter on wave release with association of shipment to a pack location.

The APIs involved are

- releaseWave()
- getShipmentDetails()

The following events are raised by the releaseWave() API:

- RELEASE_WAVE.ON_SUCCESS
- RELEASE_WAVE.SHORTAGES_DETECTED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. Information from outbound sorter regarding cartons diverted or quantity diverted can update a status value in the pipeline.

The APIs involved are:

- changeShipmentContainer()
- changeShipmentStatus()

The following events are raised by the changeShipmentContainer() API:

- CHANGE_CONTAINER.ON_SUCCESS
- CHANGE_CONTAINER_STATUS.ON_SUCCESS

The following event is raised by the changeShipmentStatus() API:

- CHANGE_SHIPMENT_STATUS.ON_SUCCESS

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Integrating with Shipping Sorters

Outbound Sorters are typically used during high volume pick, pack ship operations.

The Sterling Warehouse Management System integrates with outbound sorters as follows:

1. For pre-pick containerization, carton level information is sent after wave release. For loose items, data interfaced after post-pick containerization is completed.
2. Wave release level information is sent to sorter containing lane information.

The APIs involved are:

- releaseWave()
- getShipmentDetails()

The following events are raised by the releaseWave() API:

- RELEASE_WAVE.ON_SUCCESS
- RELEASE_WAVE.SHORTAGES_DETECTED

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

1. Information from outbound sorter regarding cartons diverted or quantity diverted can update a status value in the pipeline.

The APIs involved are:

- `changeShipmentContainer()`

The following events are raised by the `changeShipmentContainer()` API:

- `CHANGE_CONTAINER.ON_SUCCESS`
- `CHANGE_CONTAINER_STATUS.ON_SUCCESS`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Integrating with Cube-a-Scans

A cube-a-scan is typically used during inbound operations to determine the dimensions or properties of an item/SKU.

The Sterling Warehouse Management System integrates with cube-a-scan by updating the item details in the Sterling Warehouse Management System.

The API involved is:

- `manageItem()`

The following events are raised by the `manageItem()` API:

- `ITEM_DEFINITION.AFTER_MODIFY_ITEM`
- `ITEM_DEFINITION.AFTER_DELETE_ITEM`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Integrating with Weighing Scales

A weighing scale is an equipment that returns the weight of a container placed on it. Weighing scales are typically used in manifest stations for parcel shipments. For more information about setting up a weighing scale, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Configuration Guide*.

Integrating with Mettler Toledo Weighing Scales

The Sterling Warehouse Management System supports out-of-the-box integration with the Mettler Toledo PS Weighing Scale, which is compatible with various shipping systems including UPS and FedEx.

For more information about installing the Mettler Toledo Weighing Scale, see the *Sterling Selling and Fulfillment Foundation: Installation Guide*.

For more information about configuring the Mettler Toledo Weighing Scale on the Sterling Warehouse Management System, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Configuration Guide*.

Integrating with Other Weighing Scales

Additional weighing scale connectors can be built by implementing the YCPWeighingScaleConnector interface available in the package com.yantra.ycp.ui.io in the Java Archive File platform_afc.jar.

The following is a sample code for implementing the YCPWeighingScaleConnector interface:

```
public class CustomScaleConnector implements YCPWeighingScaleConnector {
    private YFCSerialIO sio;
    /* This assumes that the weighing scale is connected through serial port.
    You will need to write custom code to support other ports such as USB.*/
    private YFCPortConfig config;

    public CustomScaleConnector() {
    }

    public void init(YFCElement configEle) {
        sio = new YFCSerialIO();
        String portId = configEle.getAttribute("PortId");
        config = new YFCPortConfig(portId);
    }

    public double getWeight() {
        sio.openConnection(config);
        sio.write("W"); // command to get weight from the scale
        sio.waitForResponse(20, 1000); // sleep 20ms. every time and timeout out
        after 1 sec.
        String response = sio.read();
        return processResponse(response);
    }

    private double processResponse(String response) {
        double weight = -1;
        // process the response appropriately
        return weight;
    }

    public void resetScale() {
        // send reset command if required
    }
}
```

During initialization, the init method is called once by the YCPWeighingFactory interface.

At init time, a config XML is passed to the CustomScaleConnector. This XML is stored in the Sterling Selling and Fulfillment Foundation config database (in Device Configuration) with the class name CustomScaleConnector.

The config XML format used for the Mettler Toledo Weighing Scale is as follows:

```
<DeviceParamsXML>
  <Attributes>
    <Attribute Name="ClassName" Value="" />
    <Attribute Name="PortId" Value="" />
    <Attribute Name="BaudRate" Value="" />
    <Attribute Name="DataBits" Value="" />
    <Attribute Name="StopBits" Value="" />
    <Attribute Name="Parity" Value="" />
    <Attribute Name="FlowIn" Value="" />
    <Attribute Name="FlowOut" Value="" />
    <!-- other extended attributes specific to weighing scale
```

```
connector implementations -->
    <Attribute Name="" Value="" />
</Attributes>
</DeviceParamsXML>
```

The config XML can be configured using the Device Configuration of Type 'Weighing Scale' in the Applications Manager. For more information see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Configuration Guide*.

Note: The implementation of the YCPWeighingFactory interface must ensure that an instance can be reused across invocations. The YCPWeighingFactory interface calls `init` once during initialization, and subsequently reuses the initialized instance.

For more details about integrating the Sterling Warehouse Management System with other weighing scales, see Java Doc referring to the `com.yantra.ycp.ui.io` package.

Chapter 8. Integrating with Enterprise Resource Planning Systems

Overview of Integration with ERP Components

An Enterprise Resource Planning (ERP) system is a packaged business software system that allows a company to automate and integrate the majority of its business processes. This enables the company to share common data and practices across the entire enterprise, and to produce and access information in a real-time environment.

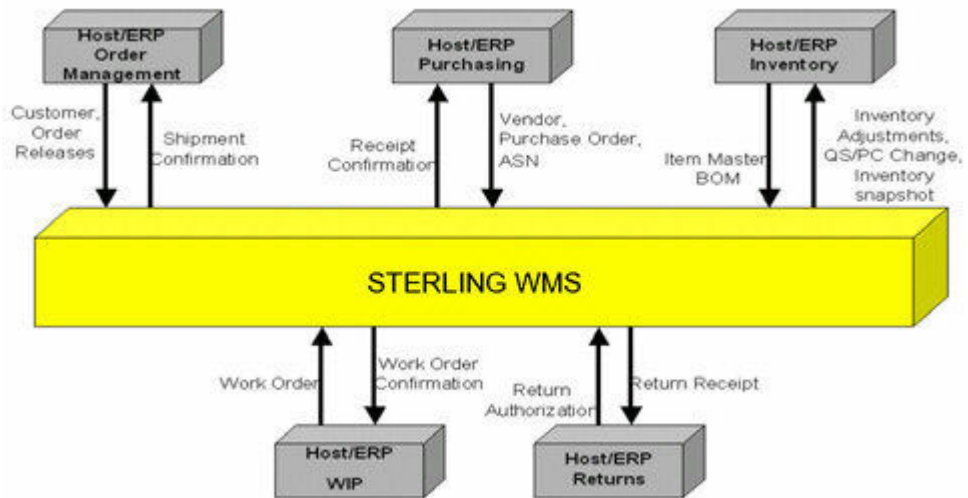
The Sterling Warehouse Management System can integrate with an ERP system to utilize any additional functions that are available in the existing environment. The Sterling Warehouse Management System can be integrated with one or more of the following components of an ERP system:

- Order Management
- Purchasing
- Inventory
- WIP
- Returns

For example, the Sterling Warehouse Management System can integrate with an ERP system to enable users to:

- Enter information in one system and ensure the accessibility and accuracy of the same information across the other application, if necessary, without duplication of data entry.
- Maintain the data entry and ownership at one point, the source module. Synchronize reference (common) data based on the static or dynamic nature of the data, and/or, as deemed necessary in a business environment.
- Perform the necessary business functions involving data sharing and transfer without having to be aware of the system links, the transfer mechanism and the programming details.
- Define and maintain the implementation setup of the integration to suit specific business needs. Typically, the user-definable parameters correspond to the modules installed, the active interfaces, frequency of data synchronization and real time or batch data transfer options.

Integration Data Flow Diagram



Integration Protocol

Sterling Selling and Fulfillment Foundation provides APIs to integrate the Sterling Warehouse Management System with ERP applications, and transfer data from an ERP system to the Sterling Warehouse Management System. These APIs can be invoked from the Service Definition Framework.

Data Exchange from an ERP System to the Sterling Warehouse Management System

Data exchange from an ERP application to the Sterling Warehouse Management System can be carried out using the Service Definition Framework in two modes:

- Asynchronous Mode (DB, JMS, MSMQ)
- Synchronous Mode (HTTP, EJB, LOCAL)

For more information about configuring these modes to facilitate integration, see the Programming Transactions chapter in the *Sterling Selling and Fulfillment Foundation: Extending Transactions*.

Data Exchange from the Sterling Warehouse Management System to an ERP System

The Sterling Selling and Fulfillment Foundation APIs raise Events, which can be configured to transfer data from the Sterling Warehouse Management System to an ERP application.

For more information about configuring Events, see the Programming Transactions chapter in the *Sterling Selling and Fulfillment Foundation: Extending Transactions*.

ERP Integration Specification: Order Management

Customer Download from an ERP System to the Sterling Warehouse Management System

Vendor information is downloaded from an ERP system to the Sterling Warehouse Management System.

The API involved is:

- `manageCustomer()`

For more information about APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Shipment/Order Release Download from an ERP System to the Sterling Warehouse Management System

Order releases or Shipment requests are downloaded from an ERP system to the Sterling Warehouse Management System.

The APIs involved are:

- `createShipment()`
- `consolidateToShipment()`

For more information about APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Shipment Confirmation Upload from the Sterling Warehouse Management System to an ERP System

Order releases or Shipment requests are uploaded from the Sterling Warehouse Management System to an ERP system.

The API involved is:

- `confirmShipment()`

The following events are raised by the `confirmShipment()` API:

- `CONFIRM_SHIPMENT.ON_SUCCESS`
- `CREATE_CONFIRM_SHIPMENT.ON_SUCCESS`
- `SHIP_SHIPMENT.ON_SHIP_CONFIRM_POST_VOID`
- `SHIP_ORDER.ON_SHIP_CONFIRM_POST_VOID`
- `INVENTORY_CHANGE.ON_CHANGE`
- `INVENTORY_COST_CHANGE.INVENTORY_VALUE_CHANGE`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

ERP Integration Specification: Purchasing

Vendor Download from an ERP System to the Sterling Warehouse Management System

Vendor information is downloaded from an ERP system to the Sterling Warehouse Management System.

The API involved is:

- `manageVendor()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Purchase Order Download from an ERP System to the Sterling Warehouse Management System

Purchase Orders are created on an ERP system and downloaded to the Sterling Warehouse Management System. PO modifications are also downloaded to the Sterling Warehouse Management System.

The APIs involved are:

- `createOrder()`
- `changeOrder()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Purchase Order Closure Download from an ERP System to the Sterling Warehouse Management System

When a PO or PO line is closed on an ERP system, it is downloaded to the Sterling Warehouse Management System.

The API involved is:

- `shortOrder()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

ASN Download from an ERP System to the Sterling Warehouse Management System

When an ASN is created on an ERP system, it can be downloaded to the Sterling Warehouse Management System.

The API involved is:

- `confirmShipment()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Receipt Upload from the Sterling Warehouse Management System to an ERP System

Receipt information can be uploaded as and when a receipt is made or when a receipt is closed.

The APIs involved are:

- closeReceipt() or
- receiveOrder()

The following event is raised by the closeReceipt() API:

- RECEIPT_COMPLETE.ON_RECEIPT_COMPLETE

The following events are raised by the receiveOrder() API:

- RECEIVE_RECEIPT.ON_SUCCESS
- RECEIVE_RECEIPT.ON_SKU_RECEIPT
- RECEIVE_RECEIPT.ON_CASE_RECEIPT
- RECEIVE_RECEIPT.ON_PALLET_RECEIPT
- INVENTORY_COST_CHANGE.INVENTORY_COST_CHANGE
- RECEIVE_ORDER.INVENTORY_COST_WRITEOFF
- RECEIVE_ORDER.INVENTORY_VALUE_CHANGE

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

ERP Integration Specification: Inventory

Item Download from an ERP System to the Sterling Warehouse Management System

New items are created on an ERP system and then downloaded to the Sterling Warehouse Management System. Typically, the ERP system is the master. However, several attributes of items required for warehouse operations are maintained in the WMS after the download of item information from the ERP system.

The API involved is:

- manageItem()

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Item Attributes Upload from the Sterling Warehouse Management System to an ERP System

Some of the item attributes, such as item dimensions and weight, can be maintained in the Sterling Warehouse Management System and then uploaded to an ERP system.

The API involved is:

- manageItem()

The following events are raised by the `manageItem()` API:

- `ITEM_DEFINITION.AFTER_MODIFY_ITEM`
- `ITEM_DEFINITION.AFTER_DELETE_ITEM`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Inventory Change Upload from the Sterling Warehouse Management System to an ERP System

Inventory changes from the Sterling Warehouse Management System are uploaded to an ERP system.

The API involved is:

- `adjustInventory()`

The following events are raised by the `adjustInventory()` API:

- `INVENTORY_CHANGE.INVENTORY_CHANGE`
- `INVENTORY_CHANGE.SUPPLY_CHANGE`
- `INVENTORY_COST_CHANGE.INVENTORY_VALUE_CHANGE`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Inventory Snapshot Upload from the Sterling Warehouse Management System to an ERP System

Inventory snapshot information may need to be uploaded from the Sterling Warehouse Management System to an ERP system.

The APIs involved are:

- `getInventoryMismatch()`
- `getInventorySnapshot()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

ERP System Integration Specification: WIP

BOM Download from an ERP System to the Sterling Warehouse Management System

Bill of Materials (BOM) information can be maintained on an ERP system and downloaded to the Sterling Warehouse Management System.

The API involved is:

- `manageItem()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Work Order Download from an ERP System to the Sterling Warehouse Management System

Work Orders can be downloaded from an ERP system to the Sterling Warehouse Management System for execution.

The API involved is:

- createWorkOrder()

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Work Order Demand Upload for Manually Created Work Orders from the Sterling Warehouse Management System to ERP

When work orders are created manually in the Sterling Warehouse Management System, work order information needs to be uploaded to an ERP system so that component items are allocated on the ERP system.

The APIs involved are:

- createWorkOrder()
- cancelWorkOrder()
- modifyWorkOrder()

The following event is raised by the createWorkOrder() API:

- CREATE_WORK_ORDER.ON_SUCCESS

The following events are raised by the cancelWorkOrder() API:

- CANCEL_WORK_ORDER.ON_SUCCESS
- CANCEL_WORK_ORDER.WORK_ORDER_ACTIVITIES_COMPLETED

The following event is raised by the modifyWorkOrder() API:

- MODIFY_WORK_ORDER.ON_SUCCESS

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Work Order Confirmation Upload from the Sterling Warehouse Management System to an ERP System

When a Work Order is confirmed, information needs to be uploaded to the ERP system indicating quantity of work order confirmed or built.

With some ERP systems, this data may not be uploaded as and when quantity built. Instead, only work order closure is uploaded to the ERP system, indicating total quantity built for the work order.

The API involved is:

- confirmWorkOrderActivity()

The following events are raised by the `confirmWorkOrderActivity()` API:

- `CONFIRM_WORK_ORDER.ON_SUCCESS`
- `CONFIRM_WORK_ORDER.WORK_ORDER_ACTIVITIES_COMPLETED`
- `CONFIRM_WORK_ORDER.LPN_ACTIVITIES_COMPLETED`
- `CONFIRM_WORK_ORDER.SKU_ACTIVITIES_COMPLETED`
- `CONFIRM_WORK_ORDER.SNO_ACTIVITIES_COMPLETED`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Close Work Order from the Sterling Warehouse Management System to an ERP System

When all quantities for a work order is completed or the remaining quantity is canceled, data needs to be published to the ERP system indicating that work order is complete.

The API involved is:

- `changeWorkOrderStatus()`

The following event is raised by the `changeWorkOrderStatus()` API:

- `CHANGE_WORK_ORDER_STATUS.ON_SUCCESS`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

ERP Integration Specification: Returns

Return Order Download from ERP to the Sterling Warehouse Management System

Return Orders are created on an ERP system and downloaded to the Sterling Warehouse Management System. Return Order modifications are also downloaded to the Sterling Warehouse Management System.

The APIs involved are:

- `createOrder()`
- `changeOrder()`

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Return Order Closure Download from an ERP System to the Sterling Warehouse Management System

When a return is closed on the host system, it is downloaded to the Sterling Warehouse Management System. Typically, one return is one receipt. Hence, when a receipt is closed, return may be marked as Closed without a separate integration from host system.

The API involved is:

- shortOrder()

For more information about the APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Receipt Upload from the Sterling Warehouse Management System to an ERP System

Typically, return information is uploaded only when receipt is closed.

The APIs involved are:

- closeReceipt() or
- receiveOrder()

The following event is raised by the closeReceipt() API:

- RECEIPT_COMPLETE.ON_RECEIPT_COMPLETE

The following events are raised by the receiveOrder() API:

- RECEIVE_RECEIPT.ON_SUCCESS
- RECEIVE_RECEIPT.ON_SKU_RECEIPT
- RECEIVE_RECEIPT.ON_CASE_RECEIPT
- RECEIVE_RECEIPT.ON_PALLET_RECEIPT
- RECEIVE_ORDER.INVENTORY_COST_CHANGE
- RECEIVE_ORDER.INVENTORY_COST_WRITEOFF
- RECEIVE_ORDER.INVENTORY_VALUE_CHANGE

Chapter 9. Point of Sale System Integration

Integrating with Point of Sale Systems

Sterling Selling and Fulfillment Foundation enables you to integrate with point of sale systems used in stores for product check-outs and returns from customers. When a sales transaction is posted to the Sterling Warehouse Management System from a point of sale (POS), the location from which inventory has to be deducted may not be known, and hence not passed. Under such circumstances, the Sterling Warehouse Management System deducts the inventory from one or more locations that are configured for the purpose of adjustment (that is, for an Adjustment Reason Code). Depending on the availability at each location, the location is appropriately adjusted and then the next location is considered, if required. If a virtual location is one of the locations in the sequence, the inventory availability at the location is not checked and such a location is allowed to go negative.

For more information about the IBM Sterling Warehouse Management System, see the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Concepts Guide*.

API Invoked During Point of Sale Integration

The API invoked during the integration of the Sterling Warehouse Management System with Point Of Sale Systems is `adjustLocationInventory()`.

This API adjusts location inventory. In point of sale systems, it is typically called with an inventory reason code associated with an adjustment sequence, without a Location ID. It can also be called with both the Location ID and the inventory reason code associated with an adjustment sequence. The transaction does not go through if the Location ID is not passed and the inventory reason code passed does not have an adjustment sequence associated with it.

If the `adjustLocationInventory` API is called with an inventory reason code associated with an adjustment sequence and the Location ID is not passed:

- Inventory is deducted consecutively from the locations or zones specified in the adjustment sequence.
- Within a zone, inventory is deducted according to the pick sequence of the locations in the zone. For locations having the same pick sequence number, inventory is deducted in the alphabetical order of the Location ID.
- Inventory in non-virtual locations is deducted only to the extent of the available quantity of loose SKU (inventory in LPN is not considered). Available inventory is deducted consecutively from the configured locations until a virtual location, if configured in the adjustment sequence, is reached. The balance of the demanded quantity is then adjusted from this virtual location. If any other locations have been configured in the adjustment sequence after the virtual location, they are ignored.
- The transaction does not go through if there is insufficient inventory in the locations or zones specified in the adjustment sequence and a virtual location has not been configured in the adjustment sequence.

When the `adjustLocationInventory` API is called with a Location ID and an inventory reason code associated with an adjustment sequence, the inventory is

adjusted in the specified location and the adjustment sequence is ignored. The transaction does not go through if there is insufficient inventory at the specified location.

When the `adjustLocationInventory` API is called for serialized items, the location sequence associated with an inventory reason code is always ignored.

- If the `adjustLocationInventory` API is called with a Location ID, inventory is deducted from that location. The transaction does not go through if the serial number is not found in the specified location.
- If the `adjustLocationInventory` API is called without a Location ID, inventory is deducted from any location where the serial number is found. The transaction does not go through if the specified serial number is not found in any location of the node.

Chapter 10. Integrating User and Item Data with External Systems

External System Integration Overview

Sterling Selling and Fulfillment Foundation enables you to integrate with external systems used to sell products, through multiple channels. This integration enables information on orders, availability, products, and customers to be passed between the external system and Sterling Selling and Fulfillment Foundation.

Triggering Data Synchronization

You can trigger synchronization of this data in three ways: near real-time, on-demand, and batch. These methods are described in the following table:

Table 9. Triggering Data Synchronization

Method	Description
Near real-time	Changes are communicated to the appropriate system as soon as they are processed. Note: This is applicable to both user and product synchronization.
On-demand	Occurs as a result of a customer manually triggering the synchronization from an external system.
Batch	Occurs at a specified time and automatically determines which items or customers need to be synchronized

Order Management Integration

The integration of Sterling Selling and Fulfillment Foundation order management with external systems enables the following:

- Order integration - Orders placed in the external system can be tracked and maintained in Sterling Selling and Fulfillment Foundation.
- Order details - When order details are viewed in the external system, they are retrieved in real time from Sterling Selling and Fulfillment Foundation.
- Order change and cancellation - Details about order changes or cancellations are communicated between systems.

APIs Invoked During Order Management Integration

The following APIs are invoked during order management integration:

- createOrder()
- changeOrder()
- getSalesOrderDetails()

For more information about these APIs, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

User and Item Synchronization

The synchronization of user and item data with an external system enables you to integrate the following:

- Users - User synchronization involves synchronizing a defined set of users, including details such as address and payment information.
- Items - Item synchronization involves synchronizing all the relevant item information.

For both users and items, services are provided to send and receive changes. These services are:

- SendItemChanges
- ReceiveItemChanges
- SendCustomerChanges
- ReceiveCustomerChanges

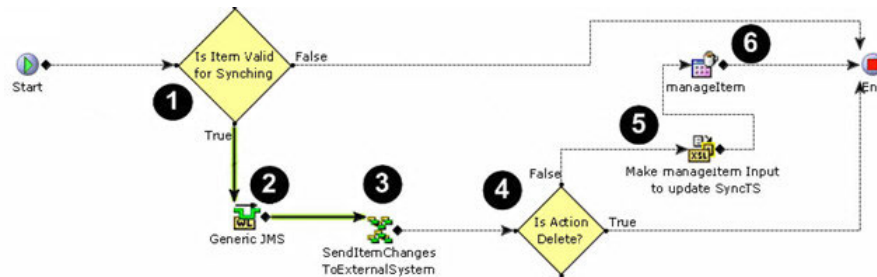
These services function by either placing or retrieving information from a JMS queue, and then passing this information to an internal or external API or service.

Item Synchronization Services in Sterling Selling and Fulfillment Foundation

Sterling Selling and Fulfillment Foundation has two main services for the synchronization of items. These services leverage APIs as well as other services in order to send or receive changes to items.

SendItemChanges Service

The sendItemChanges service is used to relay changes made to items in Sterling Selling and Fulfillment Foundation to the external system. This service is triggered as soon as an update or change to an item is made. The following figure shows the process flow of the sendItemChanges service.



The following steps are described in the figure:

Table 10. SendItemChanges Service

Step	Description
1. Is the item valid for synching?	If the item is valid for synchronization, the service continues; if it is not, the service ends. Items are deemed valid for synchronization if the ItemGroupCode is equal to PROD and the item is not a dynamic physical kit or a logical kit.

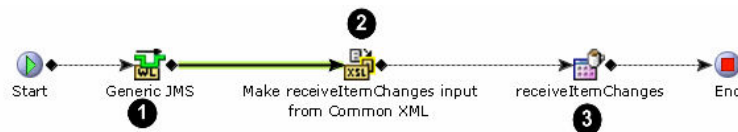
Table 10. SendItemChanges Service (continued)

Step	Description
2. Generic JMS queue	The Generic JMS queue stores messages until they can continue through the service. Note: When configuring the SendItemChanges service, the Provider URL for both the JMS Sender and JMS Receiver must be manually configured. The queue name for both must also be set to ItemSyncQueue. For more information about configuring services, see the <i>Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide</i> .
3. sendItemChangesTOExternalSystem	This service contains modules that provide an XSL translation to create a common XML file for the item, and send the XML to the external system. Note: The Java class name for the external client must be specified in the sendItemChangesTOExternalSystem service's API component.
4. Is the action a delete?	If the change being made to the item is deletion, the service ends. If it is not, the service continues.
5. Make manageItem input to SyncTS	An XSL translation takes place which adds a timestamp for when the synchronization took place. Note: SyncTS is the only column change that can occur in this XSL.
6. manageItem API	The item XML is passed to the manageItem API which commits the changes to Sterling Selling and Fulfillment Foundation.

Note: For more information about the SendItemChanges service, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

ReceiveItemChanges Service

The receiveItemChanges service accepts changes made to items in the external system and commits them to Sterling Selling and Fulfillment Foundation, if running in near-real-time mode. If batch mode is used, the service is called after the item synchronization cron job is run. The following figure shows the process flow of the receiveItemChanges service:



The following table describes the ReceiveItemChanges service:

Table 11. ReceiveItemChanges Service

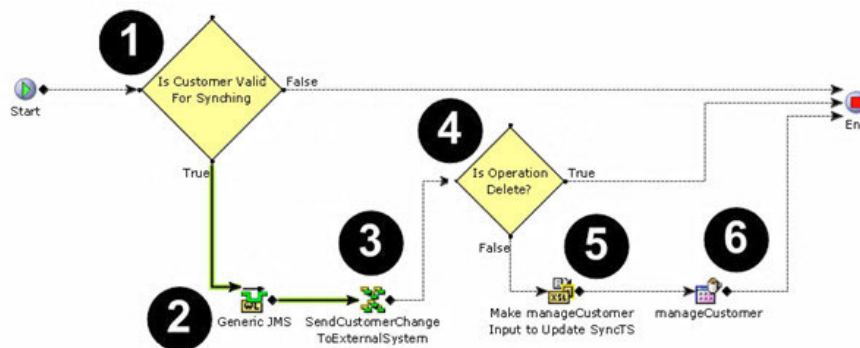
Step	Description
1. Generic JMS queue	The Generic JMS queue stores messages until they can continue through the service. Note: When configuring the ReceiveItemChanges service, the Provider URL for the JMS Receiver must be manually configured. The queue name must also be set to ItemSyncReceiveItemChangesQueue. For more information about configuring services, see the <i>Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide</i> .
2. Make receiveItemChanges input from common XML	The XSL translation takes the common XML from the external system and removes all synchronization related data as well as transforms the XML into a format that can be read by Sterling Selling and Fulfillment Foundation. Note: By default, items are deleted from Sterling Selling and Fulfillment Foundation when a message for deletion is received from the external system. This can be avoided by modifying the XSL translator in this step by changing Action="Delete" to Action="Modify" and placing the item into a custom status.
3. receiveItemChanges API	The receiveItemChanges API accepts the item XML from the external system and invokes the functionality of the manageItem API.

Customer Synchronization Services in Sterling Selling and Fulfillment Foundation

Sterling Selling and Fulfillment Foundation has two main services for the synchronization of customers. These services leverage APIs as well as other services in order to send or receive changes to customers.

SendCustomerChanges Service

The sendCustomerChanges service communicates changes made to customers in Sterling Selling and Fulfillment Foundation to the external system. The following figure illustrates a process flow of the sendCustomerChanges service:



The following table describes the sendCustomerChanges service:

Table 12. SendCustomerChanges Service

Step	Description
1. Is the customer valid for synching?	If the customer is valid for synchronization, the service continues; if it is not, the service ends. Customers are deemed valid for synchronization if they are a consumer, have a user ID that is not blank, and have an IsSyncRequired flag set to 'Y'.
2. Generic JMS queue	The Generic JMS queue stores messages until they can continue through the service. Note: When configuring the sendCustomerChanges service, the URL for both the JMS Sender and JMS Receiver must be manually configured. The queue name for both must also be set to CustomerSyncQueue. For more information about configuring services, see the <i>Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide</i> .
3. sendCustomerChangesTOExternalSystem	This service contains modules that provide an XSL translation to create a common XML file for the customer, and send the XML to the external system. Note: The Java class name for the external client must be specified in the sendCustomerChangesTOExternalSystem service's API component.
4. Is the operation a delete?	If the change being made to the customer is deletion, the service ends. If it is not, the service continues.
5. Make manageCustomer input to SyncTS	An XSL translation takes place which adds a timestamp for when the synchronization took place. Note: SyncTS is the only column change that can occur in this XSL.
6. manageCustomer API	The customer XML is passed to the manageCustomer API, which commits the changes to Sterling Selling and Fulfillment Foundation.

Note: For more information about the SendCustomerChanges service, see the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

ReceiveCustomerChanges Service

The receiveCustomerChanges service accepts changes made to customers in the external system and commits them to Sterling Selling and Fulfillment Foundation. This service is triggered as soon as an update or change to a customer is made. The following figure illustrates a process flow for the receiveCustomerChanges



service:

The following table explains more about the receiveCustomerChanges service:

Table 13. ReceiveCustomerChanges Service

Step	Description
1. Generic JMS queue	The Generic JMS queue stores messages until they can continue through the service. Note: When configuring the sendCustomerChanges service, the URL for both the JMS Sender and JMS Receiver must be manually configured. The queue name for both must also be set to CustomerSyncQueue. For more information about configuring services, see the <i>Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide</i> .
2. Make receiveCustomerChanges input from common XML	The XSL translation takes the common XML from the external system and removes all synchronization related data as well as transforms the XML into a format that can be read by Sterling Selling and Fulfillment Foundation.
3. receiveCustomerChanges API	The receiveCustomerChanges API accepts the item XML from the external system and invokes the functionality of the manageItem API.

Modifying Customer Event Templates

About this task

Manual changes to the customer event template XML files are required to enable customer synchronization. To modify the customer event template XML files:

Procedure

1. Navigate to the <OF_INSTALL_DIR>/repository/xapi/template/merged/event/ directory.
2. Locate the CUSTOMER_DEFINITION.AFTER_CREATE_CUSTOMER.xml, CUSTOMER_DEFINITION.AFTER_DELETE_CUSTOMER.xml, and CUSTOMER_DEFINITION.AFTER_MODIFY_CUSTOMER.xml files.
3. Copy the files mentioned in 2 to the <OF_INSTALL_DIR>/extensions/global/template/event directory.
4. Modify the customer event templates listed in 2 to match the common XML provided in “Customer Data Mapping” on page 113.
5. Add the following attributes to the <Customer> element in the files mentioned in 2.
 - IsSyncRequired=""
 - MaxModifyTS=""
 - SyncTS=""

Results

For more information about modifying template XML files, see the *Sterling Selling and Fulfillment Foundation: Customizing APIs*.

Data Mapping

This is the mapping that takes place during the synchronization of items and customers between an external system and Sterling Selling and Fulfillment Foundation.

Customer Data Mapping

The following common XML is used to communicate with external systems:

```
<Customer OrganizationCode="" Operation="" CustomerType="">
  <CustomerContactList >
    <CustomerContact DayFaxNo="" DayPhone="" EmailID=""
EveningFaxNo=""
EveningPhone="" FirstName="" LastName="" MobilePhone=""
Title="" UserID="">
      <CustomerAdditionalAddressList Reset="y" >
        <CustomerAdditionalAddress
CustomerAdditionalAddressID="" IsShipTo=""
IsBillTo="" IsSoldTo="" IsDefaultShipTo="" IsDefaultBillTo=""
IsDefaultSoldTo="">
          <PersonInfo AddressLine1="" AddressLine2=""
AddressLine3="" City="" Country=""
State="" ZipCode="" />
        </CustomerAdditionalAddress>
      </CustomerAdditionalAddressList>
      <CustomerPaymentMethodList Reset="Y">
        <CustomerPaymentMethod CreditCardExpDate=""
FirstName=""
MiddleName="" LastName="" CreditCardNo="" CreditCardType=""
PaymentType="" IsDefaultMethod="" />
      </CustomerPaymentMethodList>
    </CustomerContact>
  </CustomerContactList>
</Customer>
```

Database Fields for Customer Data Mapping

These are the database fields for Customer Data Mapping:

Sterling Selling and Fulfillment Foundation Database Fields
YFS_CUSTOMER_CONTACT.USER_ID
YFS_CUSTOMER_CONTACT.LAST_NAME
YFS_CUSTOMER_CONTACT.FIRST_NAME
YFS_CUSTOMER_CONTACT.TITLE
YFS_CUSTOMER_CONTACT.EMAILID
YFS_CUSTOMER_PAYMENT_METHOD.PAYMENT_TYPE
YFS_CUSTOMER_PAYMENT_METHOD.CREDIT_CARD_NO
YFS_CUSTOMER_PAYMENT_METHOD.CREDIT_CARD_EXP_DATE
YFS_CUSTOMER_PAYMENT_METHOD.CREDIT_CARD_TYPE
YFS_CUSTOMER_PAYMENT_METHOD.FIRST_NAME
YFS_CUSTOMER_PAYMENT_METHOD.MIDDLE_NAME
YFS_CUSTOMER_PAYMENT_METHOD.LAST_NAME
YFS_CUSTOMER.ORGANIZATION_CODE
YFS_PERSON_INFO.ADDRESS_LINE1

Sterling Selling and Fulfillment Foundation Database Fields
YFS_PERSON_INFO.ADDRESS_LINE2
YFS_PERSON_INFO.ADDRESS_LINE3
YFS_PERSON_INFO.CITY
YFS_PERSON_INFO.ZIP_CODE
YFS_PERSON_INFO.STATE
YFS_PERSON_INFO.COUNTRY
YFS_CUSTOMER_ADDNL_ADDRESS.IS_SOLD_TO
YFS_CUSTOMER_ADDNL_ADDRESS.IS_SHIP_TO
YFS_CUSTOMER_ADDNL_ADDRESS.IS_BILL_TO
YFS_CUSTOMER_ADDNL_ADDRESS.IS_DEFAULT_SOLD_TO
YFS_CUSTOMER_ADDNL_ADDRESS.IS_DEFAULT_SHIP_TO
YFS_CUSTOMER_ADDNL_ADDRESS.IS_DEFAULT_BILL_TO
YFS_CUSTOMER_CONTACT.<tablename>
Note: The <table_name> column is determined by the value of the CMGT_PHONES.PHONE_TYPE_CODE column. The customer phone number is then stored in this column.

Note: Extended attributes can be provided under the /Item/@Extn element.

Item Data Mapping

The following common XML is used to communicate with external systems:

```
<Item Action="Create/Modify/Delete" ItemID="" UnitOfMeasure=""
OrganizationCode="" ShortDescription="" ExtendedDescription=""
BundleFulfillmentMode="" LeadTime="" MinOrderQuantity="" IsModelItem="" Model=""
ModelItemUnitOfMeasure="" KitCode="" ConfiguredModelKey="" IsConfigurable=""
IsPreConfigured="">
  <ItemInstructionList Reset="">
    <ItemInstruction InstructionText="" SeqNo=""
InstructionType="ORDERING"/>
  </ItemInstructionList>
  <Components Reset="">
    <Component ComponentItemID="" ComponentOrganizationCode=""
ComponentUnitOfMeasure="" KitQuantity="" />
  </Components>
</Item>
```

Database Fields for Item Data Mapping

These are the database fields for the attributes for Item Data Mapping:

Attribute	Order Fulfillment Database Field	Comment
Item		
ItemID	YFS_ITEM.ITEM_ID	
UnitOfMeasure	YFS_ITEM.UOM	Note: The values in this field must be manually kept in synch between the two applications.
OrganizationCode	YFS_ITEM.ORGANIZATION_CODE	Assume that the catalog is maintained at the hub level.

Attribute	Order Fulfillment Database Field	Comment
ShortDescription	YFS_ITEM.SHORT_DESCRIPTION	This field is required to avoid errors.
Extended Description	YFS_ITEM.EXTENDED_DESCRIPTION	
BundleFulfillment Mode	YFS_ITEM.BUNDLE_FULFILLMENT_MODE	This value should be based on the following: <ul style="list-style-type: none"> • 01" for ShipTogether when the configurable item is a non-container only item. • 02" for Ship Independently when the configurable item is a container only item.
LeadTime	YFS_ITEM.LEAD_TIME	
MinOrderQuantity	YFS_ITEM.MIN_ORDER_QUANTITY	
Model	YFS_ITEM.MODEL	The existing MODEL field is used to store the parent SKU to represent the aggregate item. Do not confuse with CONFIGURED_MODEL_KEY of the configurable item.
ModelItemUnitOf Measure	YFS_ITEM.MODEL_ITEM_UOM	This field stores the unit of measure of the parent SKU.
IsModelItem	YFS_ITEM.IS_MODEL_ITEM	The value stored in this field should be "Y" if the product is of the type "Aggregate".
KitCode	YFS_ITEM.KIT_CODE	The value for this field is based on the following: <ul style="list-style-type: none"> • PK" if the product is of the type "ASSEMBLY" and the CMGT_PRODUCT.COMPONENT_SUB_TYPE field indicates that the product is a physical kit. • BUNDLE" if the product is of the type "ASSEMBLY" and the CMGT_PRODUCT.COMPONENT_SUB_TYPE field indicates that the product is a bundle. • BUNDLE" if product is of the type "CONFIGURABLE"
ConfiguredModelKey	YFS_ITEM.CONFIGURED_MODEL_KEY	
IsConfigurable	YFS_ITEM.IS_CONFIGURABLE	The value of this field should be "Y" if the product is of the type "CONFIGURABLE".

Attribute	Order Fulfillment Database Field	Comment
IsPreConfigured	YFS_ITEM.IS_PRE_CONFIGURED	
ItemInstructionList/ItemInstruction		
InstructionText	YFS_ITEM_INSTRUCTION.INSTRUCTION_TEXT	
InstructionType	YFS_ITEM_INSTRUCTION.INSTRUCTION_TYPE	
SeqNo	YFS_ITEM_INSTRUCTION.SEQ_NO	
Components/Component		
ComponentItemID	YFS_KIT_ITEM.COMPONENT_ITEM_KEY Based on ComponentItemID, ComponentOrganizationCode, and ComponentUnitOfMeasure.	
ComponentOrganizationCode	YFS_KIT_ITEM.COMPONENT_IT Based on ComponentItemID, ComponentOrganizationCode, ComponentUnitOfMeasure	Assume that the catalog is maintained at the hub level.
ComponentUnitOfMeasure	YFS_KIT_ITEM.COMPONENT_ITEM_KEY Based on ComponentItemID, ComponentOrganizationCode, ComponentUnitOfMeasure	Unit of measure of the config line (configurable) or part (assembly)
KitQuantity	YFS_KIT_ITEM.KIT_QUANTITY	

Note: Extended attributes can be provided under the /Item/@Extn element.

Note: Product item statuses must be manually kept in sync between the external system and Sterling Selling and Fulfillment Foundation.

There are two scenarios in which statuses are updated during product item synchronization:

- A new product item is added to either the external system or Sterling Selling and Fulfillment Foundation. During synchronization, the product item is added, and the status updated to Held in Sterling Selling and Fulfillment Foundation or In Creation in the external system.
- If a product item is deleted in Sterling Selling and Fulfillment Foundation, the status in the external system is updated to Blocked. In addition, when that product item is retrieved in the UI of the external system, text is displayed in the UI to indicate that this product has been deleted in Sterling Selling and Fulfillment Foundation.

- If a product is deleted from the external system, a message is sent to Sterling Selling and Fulfillment Foundation. By default, items are removed from the database. This can be avoided by modifying the XSL translator in this step by changing `Action="Delete"` to `Action="Modify"` and placing the item into a custom status.

Chapter 11. Integrating with JMS Systems

Introduction to Integrating with JMS

In order for some service nodes to communicate with external applications, external message queueing software must be configured. The following third-party message queueing applications are supported:

- Oracle WebLogic JMS
- IBM WebSphere MQ
- IBM WebSphere Default Messaging
- JBoss Messaging JMS
- TIBCO JMS

Configuring Oracle WebLogic JMS

About this task

For information specific to using WebLogic, see the documentation provided by Oracle.

To configure WebLogic JMS:

Procedure

1. Invoke the WebLogic console by entering the URL for Application Consoles. For example, `http://<IP address of machine where weblogic is installed>:<port>/console`.
2. Log in as Administrator.
3. In the left-hand panel, click Services > JDBC > Connection Pools.
4. If message persistence or paging is required, right-click Connection Pools and choose configure a new JDBCConnectionPool.
5. Configure the new JDBC pool with the following values:
 - Name - Any name, for example, MyJDBCPool
 - URL - `jdbc:oracle:thin:@<IPAddress>:1521:<SID>`
 - DriverClassName - `oracle.jdbc.OracleDriver`
 - Properties -
 - `user=<username>`
 - `password=<password>`
6. Select the Targets tab. In the left-hand panel, select one or more servers. (Several choices may appear if your server is in a clustered environment.) Then click the right arrow button to move the servers you have selected to the panel on the right.
7. In the left-hand panel, right-click JMS > ConnectionFactories to configure a new Connection Factory.

The JNDIName must match the QCFlookup value in the Applications Manager for the WebLogic JMS Transport Type.

8. Select the Targets tab. In the left-hand panel, select one or more servers. (Several choices may appear if your server is in a clustered environment.) Then click the right arrow button to move the selected server to the window on the right.
9. If message persistence or paging is required, right-click Stores, and configure a new JMSJDBCStore or Filestore.
 - a. If you choose JDBCStore, using the Connection Pool drop-down list, select your connection pool.
 - b. Right-click Servers and configure a new JMS server.
 - c. Select the store from the drop-down list.
10. Select the Targets tab. In the left-hand window, select *one* server. (Several choices may appear if your server is in a clustered environment; you can select only one of them.). Then click the right arrow button to move the selected server to the window on the right.
11. Within the newly configured JMS server, click Destinations and configure all required JMS Queues. Now all of the JMS queues are configured.
When configuring services that use WebLogic JMS, use the JNDI Name value from the WLS configuration as the message queue name.
12. Restart the WebLogic server for these new settings to take effect.
13. Launch the integration server by running `startIntegrationServer.sh` (or `.cmd`) in `<INSTALL_DIR>/bin`.
14. If you need to run multiple servers, repeat 13 for each additional server.

WebLogic Time-Out Considerations for Transacted Sessions

When using WebLogic JMS as a messaging system to receive messages in transactional mode and no messages are received for a period of time equal to the WebLogic transaction time-out value (defaults to 3600 seconds), the following error message appears in the integration server. After this error message appears, no messages can be processed and you must relaunch the adapter in order to process any messages that recently arrived.

```
<date-time> [Thread-6] ERROR services.jms.JMSConsumer -Could not successfully
process message
weblogic.jms.common.TransactionRolledBackException:
  at weblogic.rmi.internal.BasicOutboundRequest.sendReceive
    (BasicOutboundRequest.java:85)
  at weblogic.rmi.internal.BasicRemoteRef.invoke(BasicRemoteRef.java:135)
  at weblogic.rmi.internal.ProxyStub.invoke(ProxyStub.java:35)
  at $Proxy2.dispatchSyncNoTranFuture(Unknown Source)
  at weblogic.jms.dispatcher.DispatcherWrapperState.dispatchSyncNoTran
    (DispatcherWrapperState.java:341)
  at weblogic.jms.client.JMSSession.receiveMessage(JMSSession.java:347)
  at weblogic.jms.client.JMSConsumer.receive(JMSConsumer.java:333)
  at weblogic.jms.client.JMSConsumer.receive(JMSConsumer.java:279)
  at com.yantra.interop.services.jms.JMSConsumer.run(JMSConsumer.java:204)
  at java.lang.Thread.run(Thread.java:512)
```

For help with choosing an appropriate transaction time-out value for your system, see your WebLogic documentation.

Before You Begin Configuring IBM WebSphere MQ

For information specific to using WebSphere MQ, see the documentation provided by IBM.

These directions assume that the following have been successfully installed:

- WebSphere MQ software
- WebSphere MQ Java classes
- WebSphere MQ JMS support pack

Creating the Queue Manager and Queues for IBM WebSphere MQ

About this task

In Windows, you can use the MQSeries® Explorer, or follow the directions below.

To create the Queue Manager and Queues:

Procedure

1. Log in as the WebSphere MQ user or as a user belonging to the mqm user group.
2. Navigate to the directory where WebSphere MQ has been installed. Typically the location is as follows:
 - If you are using UNIX - /opt/mqm/bin
 - If you are using Windows - <WebSphere MQ Install Directory>\bin
3. Run the dspmq command to find out which queue managers, if any, exist.
 - If a suitable queue manager exists, start it using the strmqm <qmgr> command. The queue manager can be stopped by using the endmqm <qmgr> command.
 - If no queue manager exists, use the crtmqm <MYQMGR> command to create one.
4. Run the runmqsc command to send commands for creating queues. For examples of these commands, see below:

```
runmqsc MYQMGR
DEFINE QLOCAL ('getATP');
DEFINE QLOCAL ('createOrder');
END
```

Note: WebSphere MQ converts all characters to upper case, which causes errors. To use mixed case names, enclose them within single quotation marks, for example, DEFINE QLOCAL ('getATP').

Configuring a Queue Manager to Client Connection for IBM WebSphere MQ

In order to send messages to a WebSphere MQ queue on another computer, the QManager must be configured for the server and the client computer.

When a new queue is created in WebSphere MQ, the following default values are assigned to it:

- MAXDEPTH - Maximum number of messages that a queue can hold. Defaults to 5000.

- MAXMSGL - Maximum size of a message. Defaults to 4 MB.

These settings may need to be adjusted depending on the load and speed of the third-party application that submits the messages, as opposed to the third-party application that retrieves the messages.

Creating JMS Bindings in IBM WebSphere MQ Procedure

1. On the server computer, create a QueueManager <QManagerName>.
2. On the server computer's command line, run the following executable:

```
<MQInstallDir>/bin/runmqsr -m <QManagerName> -t TCP -p <PORT>
```
3. On the client computer, edit the JMSAdmin.config properties file to contain the following lines:

```
INITIAL_CONTEXT_FACTORY=<JNDI_ICF>
PROVIDER_URL=<JNDI_URL>
```

where <JNDI_ICF> is the Initial Context Factory (ICF) class for use with the JNDI you have chosen. For example, `com.sun.jndi.fscontext.RefFSContextFactory`. <JNDI_URL> is the path of the provider URL which is provided in the format expected by the JNDI server and ICF.

4. On the client computer, create a .scp command file that contains the following parameters:

```
def qcf( <QCFName> ) qmgr(<QManagerName>) transport(CLIENT) host(<ipaddress
of Server> ) channel(SYSTEM.DEF.SVRCONN) port( <PORT> )
def q(getATP) qu(getATP)
def q(reply_getATP) qu(reply_getATP)
def q(createOrder) qu(createOrder)
end
```

5. On the client computer, pass the .scp file to the WebSphere MQ JMSAdmin class using the following syntax:

```
java com.ibm.mq.jms.admin.JMSAdmin < intsetup.scp
```

This creates a .bindings file in the directory specified for the provider URL. All the JAR files in <MQ_HOME>/java/lib/ directory should be listed in your CLASSPATH environment variable.

Removing JMS Bindings in IBM WebSphere MQ About this task

To unbind the queues from JNDI

Procedure

Create a .scp command file and pass it into the WebSphere MQ JMSAdmin program.

The following are example commands:

```
del qcf(ivtQCF)
del q('getATP')
del q('reply_getATP')
del q('createOrder')
end
```

Archive Files

Since the example configuration uses the client transport, the `com.ibm.mqbind.jar` file is not necessary. However, the client does use the following MQ-specific JAR files:

- /mqclient/java/lib/com.ibm.mq.jar
- /mqclient/java/lib/com.ibm.mqjms.jar
- /mqclient/java/lib/connector.jar
- /mqclient/java/lib/fscontext.jar
- /mqclient/java/lib/jms.jar
- /mqclient/java/lib/jndi.jar
- /mqclient/java/lib/jta.jar
- /mqclient/java/lib/providerutil.jar

Configuring Sterling Selling and Fulfillment Foundation to Use WebSphere MQ Queues

About this task

When configuring Sterling Selling and Fulfillment Foundation to use the WebSphere MQ queues, see the WebSphere MQ node in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

To configure a service definition:

Procedure

1. Log in to Sterling Selling and Fulfillment Foundation as the user who belongs to the mqm user group (otherwise, the WebSphere MQ adapter does not launch).
2. Use the Applications Manager to configure the service. While configuring a WebSphere MQ service, enter the following:
 - The initial Context Factory `com.sun.jndi.fscontext.RefFSContextFactory`, and
 - A provider URL as `file:/<pathOfTheProviderURL>`

Note: The values for the Context Factory and the Provider URL must match those in the `JMSadmin.config` file.

3. Launch the integration server by running `startIntegrationServer.sh` (or `cmd`) in `<INSTALL_DIR>/bin`.
4. If you need to run multiple servers, repeat 3 for each additional server.

Configuring IBM WebSphere MQ for Access by WebSphere's JNDI Namespace

About this task

You can configure the WebSphere MQ queues for access by WebSphere's JNDI namespace rather than the typical file URL.

For information about the version of WebSphere MQ which includes MQ JMS client software, see *Sterling Selling and Fulfillment Foundation: Installation Guide*.

If needed, see the IBM Technical Tip "*Setting up MQ Java Message Service (JMS) Support in WebSphere Application Server*".

To configure WebSphere MQ:

Procedure

1. You should set the shared library path for UNIX and LINUX systems as follows:

```
set <Shared_Library_Path_Name>=<mqjava_install_path>/lib
```

where the <Shared_Library_Path_Name> is the shared library path environment variable for your operating system. For example:

- In AIX[®] it is LIBPATH.
- In HP-UX it is SHLIB_PATH.
- In Sun and Linux it is LD_LIBRARY_PATH.

2. Modify the <mqjava_install_path>/bin/JMSAdmin.config file as follows:

```
INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory  
PROVIDER_URL=CORBAloc://<WAS_admin_IP_address>:<WAS_bootstrap_port>
```

3. Create an ivtsetup.scp command file that contains the following parameters:

```
def qcf( <QCFName> ) qmgr(<QManagerName>) transport(CLIENT) host(<ipaddress  
of Server> ) channel(SYSTEM.DEF.SVRCONN) port( <PORT> )  
def q(JNDINameOfQueue) qu(QueueName)
```

In the following example, a QueueConnectionFactory is created with the JNDI name ivtQCF. This QueueConnectionFactory is configured to access the Queue Manager SYSTEM.TEST. Using the 'CLIENT' (network based) transport on the computer 127.0.0.1, through port 1414 (WebSphere MQ default), through the server connection channel named SYSTEM.DEF.SVRCONN (WebSphere MQ default).

Next, a queue object is created with the JNDI name getATP, which is configured to work with the getATP queue on QueueManager SYSTEM.TEST. (Of course, you must ensure that you have created this queue on the queue manager as well.)

Finally, an end command is issued to shut down JMSAdmin.

Note that the .scp file can have any name, but the convention is ivtsetup.scp (ivt=installation verification test).

```
def qcf(ivtQCF) qmgr(SYSTEM.TEST) transport(CLIENT) host(127.0.0.1)  
CHANNEL(SYSTEM.DEF.SVRCONN) port (1414)  
  
def q(getATP) qu(getATP) QMGR(SYSTEM.TEST)  
end
```

4. Set the PATH and CLASSPATH in the JMSAdmin script as follows:

```
MQJAVA_PATH=<path to ma88 installation>  
PATH=$MQJAVA_PATH  
CLASSPATH=$MQJAVA_PATH/lib:$MQJAVA_PATH/lib/com.ibm.mq.jar:$MQJAVA_PATH/lib/  
com.ibm.mqjms.jar:$MQJAVA_PATH/lib/jms.jar
```

For information about WebSphere JARs, see IBM documentation

5. Pass the .scp file to the WebSphere MQ JMSAdmin class using the following syntax:

```
java com.ibm.mq.jms.admin.JMSAdmin < intsetup.scp
```

Inside the Applications Manager

Configure a service that contains a WebSphere MQ node. Ensure that the link properties of the node match the Initial Context Factory, Provider URL, and the JNDI name specified for the desired queue.

The WebSphere MQ and WebSphere JAR files are also required for the IntegrationAdapter program and whatever client is putting the messages into the queue(s).

Inside the WebSphere Admin Console

In order to put messages into the WebSphere MQ queues from inside Sterling Selling and Fulfillment Foundation, as the Release agent needs to do or for services invoked by Actions and Events, follow the instructions provided in the *IBM Technical Tip "Configuring MQ JMS support in the WebSphere J2EE Environment"*.

If you are running on an IBM AIX system, include the following line in the script that launches the IntegrationAdapter:

```
export LDR_CNTRL=MAXDATA=0x30000000
```

Configuring IBM WebSphere Default Messaging

These directions assume that the following have been successfully installed:

- WebSphere Application Server with support for Default Messaging
- WebSphere Default Messaging Java classes
- WebSphere Default Messaging support pack

For more information specific to using the WebSphere Default Messaging, see the documentation provided by IBM.

Configuring Sterling Selling and Fulfillment Foundation to Use WebSphere Default Messaging

About this task

When configuring Sterling Selling and Fulfillment Foundation to use the WebSphere Default Messaging queues, see the WebSphere Default Messaging Queue section in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

For information about the version of WebSphere Default Messaging that includes the Default Messaging client software, see *Sterling Selling and Fulfillment Foundation: Installation Guide*.

To configure WebSphere Default Messaging:

Procedure

1. Set the shared library path for UNIX and LINUX system as follows:
set <Shared_Library_Path_Name>=<dm_java_install_path>/lib
where <Shared_Library_Path_Name> is the shared library path environment variable for your operating system.
For example:
 - On AIX it is: LIBPAT
 - On Linux it is: LD_LIBRARY_PATH
2. If Agent or Integration Servers communicate with the WebSphere Default Messaging, install the WAS client. The WAS client needs to be the exact version, including fix pack as the WAS server.
3. Add the following command to the startIntegrationServer.sh script prior to the java line:
\${WAS_CLIENT_HOME}/bin/setupClient.sh
where \${WAS_CLIENT_HOME} is the installation location of the WAS client.

4. Edit the startIntegrationServer.sh script to run the Agent or Integration Server to add the following system property:
`-Djava.ext.dirs=$WAS_EXT_DIRS $SERVER_ROOT $CLIENTSAS.`
5. Ensure that the following changes are made to the startIntegrationServer.sh (or cmd) startup script located in the <INSTALL_DIR>/bin directory to include the changes made in 3 on page 125 and 4.

```

WAS_CLIENT_HOME=<path of where WAS client installation>
export WAS_CLIENT_HOME
${WAS_CLIENT_HOME}/bin/setupClient.sh
java -Djava.ext.dirs=$WAS_EXT_DIRS $SERVER_ROOT $CLIENTSAS
${BOOTCLASSPATH} ${JAVA_OPTIONS} -cp ${CLASSPATH}
com.yantra.integration.adapter.IntegrationAdapter "$1"

```

JBoss Messaging JMS

Creating Queues in JBoss Messaging JMS

About this task

To create a Queue in JBoss Messaging JMS:

Procedure

Edit the <JBOSS_HOME>/server/<SERVER_NAME>/deploy/jboss-messaging.sar/destination_service.xml file to configure a queue. The following table provides a list of attributes to use to configure a queue.

Table 14. JBoss JMS Attributes

Attribute	Description
DestinationManager	Specify the object name of the DestinationManager where the queue is deployed.
SecurityManager	Specify the object name of the SecurityManager where the SecurityConf is deployed.
SecurityConf	Specify the configuration interpreted by the SecurityManager.
JNDIName	Specify the JNDI binding of the queue. If you specify none, the system looks for a jmx attribute "name" in the queue's object name.
MaxDepth	Specify the maximum depth of the queue.
InMemory	When set to true, messages are not persisted. It also avoids message softening when NullPersistenceManager is used.
RedeliveryLimit	Specify the maximum number of times a message must not be acknowledged before it is sent to DLQ. Valid values are: <ul style="list-style-type: none"> • 0 - indicates do not redeliver • n - indicates redeliver n times
RedeliveryDelay	Specify the time (in milliseconds) to wait before a message is redelivered after it is not acknowledged.
MessageCounterHistory DayLimit	Specify the number of days you want to keep the MessageCounter history.
ReceiversImpl	Specify the class you want to use for the receivers implementation.

Table 14. JBoss JMS Attributes (continued)

Attribute	Description
RecoveryRetries	Specify the recovery retries for the queue. By default, the value is set to 0 (zero). Specifies the number of times uncommitted transactions must be resolved before failing.

Sample Code for Queue Configuration

The following is a sample code for queue configuration:

```
<mbean code="org.jboss.jms.server.destination.QueueService"
  name="jboss.messaging.destination:service=Queue,name=testQueue"
  xmbean-dd="xsdesc/Queue-xmbean.xml">
  <depends optional-attribute-name="ServerPeer">jboss.messaging:service=
  ServerPeer</depends>
    <depends>jboss.messaging:service=PostOffice</depends>
    <attribute name="SecurityConfig">
      <security>
        <role name="guest" read="true" write="true"/>
        <role name="publisher" read="true" write="true" create="false"/>
        <role name="noacc" read="false" write="false" create="false"/>
      </security>
    </attribute>
  </mbean>
```

Configuring Sterling Selling and Fulfillment Foundation to Use JBoss Messaging Queues

About this task

When configuring Sterling Selling and Fulfillment Foundation to use JBoss Messaging queues, see the section about the JBoss Messaging node in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

Note: You must install JBoss messaging for using JBoss messaging queues. JBoss Messaging is supported with JBoss 4.2.0 or higher. However, for earlier versions of JBoss, only JBoss MQ is supported as a JMS.

To configure a service definition:

Procedure

1. Log in to Sterling Selling and Fulfillment Foundation as an admin user.
2. Use the Applications Manager to configure the service. While configuring a Generic JMS service, enter the following:
 - The initial Context Factory `org.jnp.interfaces.NamingContextFactory`, and
 - A provider URL as `jnp://<IP address and port of the JBoss instance>`
3. Set up the CLASSPATH for the `startIntegrationServer` script by adding the required jars to the CLASSPATH. For more information about setting up the classpath, see the section on Setting Up the Classpath for the Runtime Utilities in *Sterling Selling and Fulfillment Foundation: Installation Guide*.
4. Launch the integration server by running `startIntegrationServer.sh` (or `cmd`) in `<INSTALL_DIR>/bin`.
5. If you need to run multiple servers, repeat 4 for each additional server.

Configuring TIBCO JMS

TIBCO JMS can be configured as the messaging system for Sterling Selling and Fulfillment Foundation. For information about using TIBCO JMS, see the documentation provided by TIBCO.

TIBCO JMS Attributes

The `tibemsadmin` command is a command line utility used to create JMS objects, and to configure user and group permissions in TIBCO. This tool is available in the `<TIBCO>/ems/5.0/bin` directory.

The following table provides a list of attributes used to create JMS objects, configure users, groups, and permissions in the TIBCO Enterprise Messaging Server using the `tibemsadmin` command.

Table 15. TIBCO JMS Attributes

Attribute	Description
<code>tibemsd</code>	The command used to start TIBCO server.
<code>tibemsadmin</code>	The command used to start the TIBCO admin tool.
<code>create factory <Connection Factory Name> <Connection Factory Type></code>	The command used to create a Queue Connection Factory. Example: <code>create factory secureqcf queue</code>
<code>addprop factory <Connection Factory Name> url=<url-string></code>	The command used to set up an URL to listen to an external address such that the Queue Connection Factory is accessible from other hosts. Example: <code>addprop factory secureqcf url=tcp://devhost:7222</code>
<code>addprop queue <queuename> <propertyname>=<value></code>	Enables you to add a property to an existing queue. Example: <code>addprop queue myqueue prefetch=1</code> Where: <ul style="list-style-type: none">- the queue is called myqueue- the property is prefetch- the property value to assign is "1"
<code>create queue <queue-name></code>	The command used to create a queue. Example: <code>create queue scurequeue</code> . Note: When creating a queue, you can specify properties to be set as part of the command by adding <code><propertyname>=<value></code> after the queue name. Example: <code>create queue securequeue prefetch=1</code>
<code>create user <username></code>	The command used to create a user. Example: <code>create user secureuser</code>
<code>set password <username> <new password></code>	The command used to create a password for a user. Example: <code>set password secureuser securepassword</code> .
<code>create group <groupname></code>	The command used to create a group. Example: <code>create group securegroup</code>
<code>add member <group name> <user to be added></code>	The command used to add a user to a group. Example: <code>add member securegroup secureuserser</code>

Table 15. TIBCO JMS Attributes (continued)

Attribute	Description
grant queue <queue-name> group = <group-name> <permission>	The command used to set the requisite permissions on a queue for a group. Example: grant queue securequeue group=securegroup send grant queue securequeue group=securegroup receive grant queue securequeue group=securegroup browse
addprop queue <queue-name> secure	The command used to enable authorization for a queue. Example: addprop queue securequeue secure
setprop queue <queue-name> prefetch=1	If queue name is myQueue, then the command is setprop queue myQueue prefetch=1

Configuring TIBCO JMS as an Agent Queue

When configuring a TIBCO JMS for an agent queue, you must set the prefetch parameter for the queue to "1." If you fail to set the prefetch parameter to 1 for an agent queue, it will cause agents to attempt to process messages more than once, resulting in job execution failures and wasted processing power. You may also see multiple trigger messages in queues causing severe performance degradation.

To set the prefetch parameter, use either the create queue command or the addprop command. See "TIBCO JMS Attributes" on page 128 for details on using the commands.

Requirements for Configuring a JMS Client on TIBCO

The following parameters must be configured to connect to an unsecured queue on TIBCO:

- **URL:** tcp://<hostname>:<port>
Example: tcp://tibserver:7222
- **ICF:** com.tibco.tibjms.naming.TibjmsInitialContextFactory
- **QCF:** The name of the Queue Connection Factory created during setup.

If you need to connect to a secured queue, you must pass a username and password for both JNDI and "queuebased" security.

The following properties must be configured for use with the Service Definition Framework (SDF):

- sci.queuebasedsecurity.userid
- sci.queuebasedsecurity.password
- java.naming.security.principal
- java.naming.security.credentials

Configuring the Sterling Selling and Fulfillment Foundation To Use TIBCO Messaging Queues

About this task

When configuring the Sterling Selling and Fulfillment Foundation to use TIBCO Messaging queues, see the section about the TIBCO Messaging node in the *Sterling Selling and Fulfillment Foundation: Application Platform Configuration Guide*.

To configure a service definition:

Procedure

1. Log in to Sterling Selling and Fulfillment Foundation as an admin user.
2. Use the Applications Manager to configure the service. While configuring a Generic JMS service, enter the following:
 - The initial Context Factory
`com.tibco.tibjms.naming.TibjmsInitialContextFactory`, and
 - A provider URL as `tcp://<IP address and port of the TIBCO instance>`
3. Set up the CLASSPATH for the `startIntegrationServer` script by adding the required jars to the CLASSPATH. For more information about setting up the classpath, see the section on Setting Up the Classpath for the Runtime Utilities in *Sterling Selling and Fulfillment Foundation: Installation Guide*.
4. Launch the integration server by running `startIntegrationServer.sh` (or `cmd`) in `<INSTALL_DIR>/bin`.
5. If you need to run multiple servers, repeat 4 for each additional server.

Chapter 12. Integrating with Financial Systems

Requirements for Financial System Integration

To use the data captured using the Sterling Selling and Fulfillment Foundation Inventory Cost Management feature with your financial system, you must load the initial inventory cost data and configure process-specific events.

Load Initial Inventory Cost Data

Sterling Selling and Fulfillment Foundation provides an API to load the initial inventory value of an item at a ship node for a given quantity. The `loadInventoryNodeCost` API supports multiple items to be given in the input with inventory cost data for each ship node under that.

The `loadInventoryNodeCost` API validates the Quantity passed with the actual inventory supply information available for that item/ship node. This API only considers the supply types which are specified as on-hand and cost maintained. For more information about the input XML attributes, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

This API is called for the initial load of cost data at system start up time. This API should not be used after going into production with the Inventory Costing Management feature implemented.

The following query can be run to get the initial onhand supply quantity:

```
SELECT B.ORGANIZATION_CODE, B.ITEM_ID, B.UOM, B.PRODUCT_CLASS, A.SHIPNODE_KEY
SHIP_NODE, SUM(QUANTITY) QUANTITY
FROM YFS_INVENTORY_SUPPLY A, YFS_INVENTORY_ITEM B
WHERE A.INVENTORY_ITEM_KEY = B.INVENTORY_ITEM_KEY
AND SUPPLY_TYPE IN (
SELECT SUPPLY_TYPE FROM YFS_INVENTORY_SUPPLY_TYPE
WHERE ONHAND_SUPPLY = 'Y' AND COSTING_REQUIRED = 'Y')
GROUP BY B.ORGANIZATION_CODE, B.ITEM_ID, B.UOM, B.PRODUCT_CLASS, A.SHIPNODE_KEY
```

Configuring Process-Specific Events

In order to interface with your financial system and use the Sterling Selling and Fulfillment Foundation Inventory Costing data, you must configure the applicable events for the processes described in this section.

Receipt Process

From the General Process Type, configure the following two receipt process events for the `INVENTORY_COST_CHANGE` Transaction ID.

Receipt Process - `INVENTORY_COST_CHANGE` When Is This Event Raised?

This event is raised for any order receipt such as a purchase order, return order and so on. For example, at the time of purchase order receipt this event is raised from the inventory management module for each receipt line containing details of a single receipt line to generate G/L level postings in a financial application. One

event is published for each purchase order line as a receipt is recorded against it. If a purchase order line is received in multiple receipts, multiple events are raised.

For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update general ledger entries for accounts payable accruals and inventory value accounts.

Receipt Process - INVENTORY_COST_WRITEOFF When Is This Event Raised?

When doing a receipt against an item or node that has a negative on-hand balance, Inventory Value and Average Cost calculations need to be modified. The application generates this second event to accompany the standard inventory cost change event (*INVENTORY_COST_CHANGE*). This second event publishes the delta between recalculated inventory value and the write off amount details. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update general ledger entries for variance and inventory value accounts.

Sales Order Creation Process

The unit cost for an order line is stored as the unit cost stored for the item master. If the unit cost was manually entered at the item level in the product master tables, the order line uses the manually entered unit cost. If no manual entry was made, the order line uses the computed unit cost stored at the item level. If no such cost was stored, the cost is reflected as \$0.00 on the sales order line and the *ORDER_CREATE.ON_ZERO_UNIT_COST* event is triggered.

If the item definition is not stored in Sterling Selling and Fulfillment Foundation, the *getItemDetails* user exit may be implemented to return unit cost from an external source. For more information about the *getItemDetails* user exit, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Shipment Confirmation Process

From the General Process Type, configure the following event for the *INVENTORY_COST_CHANGE* Transaction ID.

Shipment Confirmation Process - INVENTORY_VALUE_CHANGE When Is This Event Raised?

When a sales order is shipped this event is raised from the inventory management module for each shipment line with the inventory value change information for the fulfillment location. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update general ledger entries for cost of goods sold, inventory value, and variance accounts.

Invoice Process

Using the CREATE_ORDER_INVOICE.0003 Transaction ID for returns or the CREATE_SHIPMENT_INVOICE.0001 Transaction ID for shipments, configure the following event for the Invoice process.

Invoice Process - ON_INVOICE_CREATION When Is This Event Raised?

During invoice creation, this event is raised for each invoice created. This event publishes the details about the invoice created. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This can be used to post sales and account receivables general ledger entries.

Work Order Confirmation Process

From the General Process Type, configure the following three work order confirmation process events for the INVENTORY_COST_CHANGE Transaction ID.

Work Order Confirmation Process - INVENTORY_COST_CHANGE When Is This Event Raised?

During work order processing, when the production of a kit parent item is reported, this event is raised from the inventory management module for the parent with the inventory cost change information for the production location. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update general ledger entries on the financial system.

Work Order Confirmation Process - INVENTORY_COST_WRITEOFF When Is This Event Raised?

When reporting production of a kit parent item that has a negative on-hand balance at the production location, Inventory Value and Average Cost calculations need to be modified. The application generates this second event to accompany the standard inventory cost change event (*INVENTORY_COST_CHANGE*). This second event publishes the delta between recalculated inventory value and the write off amount details. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update general ledger entries on the financial system.

Work Order Confirmation Process -INVENTORY_VALUE_CHANGE When Is This Event Raised?

When reporting production of a kit, this event is raised for each kit component. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update general ledger entries for cost of goods sold, inventory value, and variance accounts.

Inventory Adjustment Process

From the General Process Type, configure the following event for the INVENTORY_COST_CHANGE Transaction ID.

Inventory Adjustment Process - INVENTORY_VALUE_CHANGE When Is This Event Raised?

When an inventory adjustment is done for an item at a fulfillment location, this event is raised from the inventory management module with the inventory value change information for the fulfillment location. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update variance and inventory value accounts in the financial system.

Return Order Process

From the General Process Type, configure the following event for the INVENTORY_COST_CHANGE Transaction ID.

Return Order Process - INVENTORY_VALUE_CHANGE When Is This Event Raised?

At the time of return order receipt, this event is raised from the inventory management module for each return receipt line with the inventory value change information for the return location. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update variance and inventory value accounts in the financial system.

Callback from Financial System for Inventory Value Adjustment

This interface is implemented as a call to the `updateInventoryCost` API in Sterling Selling and Fulfillment Foundation. This should be used whenever the Accounts Payable application generates a variance between expected PO cost and the actual cost on the Payables Invoice. The variance amount should be passed back to

Sterling Selling and Fulfillment Foundation to be reflected in the inventory value. Sterling Selling and Fulfillment Foundation then tries to adjust the inventory value and re-compute the average cost. If the total on-hand is less than the purchase quantity (due to subsequent shipments or issues), the total variance is prorated and applied to the remaining on-hand inventory. An additional event is raised to adjust the difference in the financial system. For more information about the input attributes for the interface, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What are the Expected Updates on Sterling Selling and Fulfillment Foundation?

Inventory value is adjusted by the variance amount. Average cost is recomputed. If the total on-hand is less than what has to be adjusted, the total variance is prorated and applied on the remaining on-hand inventory. The amount not applied is passed back to the financial application so that it can be stored in an appropriate variance account.

Using the INVENTORY_COST_UPDATE Transaction ID, configure the following event for the Callback from Financial System process.

Callback from Financial System Process - COULD_NOT_APPLY_INV_VALUE_CHANGE When Is This Event Raised?

The amount not applied on Sterling Selling and Fulfillment Foundation is passed back to the financial application by raising this event which publishes the variance amount details. For more information about the data published by the event, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

What Are the Expected Updates on the Financial System?

This event can be used to update the appropriate variance account on the financial system.

Chapter 13. Rapid Deployment Features

Rapid Deployment Feature Overview

The Sterling Selling and Fulfillment Foundation Rapid Deployment Tool (RDT) can be utilized for the rapid deployment of Sterling Selling and Fulfillment Foundation.

The rapid deployment features include:

- Interface Field Mapping Documents
- Initial Data Loading

In addition to these rapid deployment features, Sterling Selling and Fulfillment Foundation provides a mechanism to create a new IBM Sterling Warehouse Management System node from an existing node.

For more information about Copying an Existing Node to a New Node, Onboarding an Enterprise to a Node, Offboarding an Enterprise from a Node, and Deleting the Current Node, refer to the *Sterling Selling and Fulfillment Foundation: Warehouse Management System Configuration Guide*.

Interface Field Mapping Documents

An Interface Field Mapping Document specifies integration mapping between Sterling Selling and Fulfillment Foundation and an external system. Typically, it is a Microsoft Excel document based on the input and output XMLs of the Sterling Selling and Fulfillment Foundation APIs or custom APIs written at the implementation phase of a project.

This feature describes the methodology to generate a Microsoft Excel-compatible XML spreadsheet file from the input/output XML file of an API, which can be used to create the Interface Field Mapping Document.

Note: The Interface Field Mapping Template generation tool can only be used in Microsoft Windows environment.

Generating Interface Field Mapping Template Documents

Sterling Selling and Fulfillment Foundation provides a tool to generate Interface Field Mapping Template documents from input/output XMLs.

The input XML for this generation could be an Input/Output XML from a Sterling Selling and Fulfillment Foundation-exposed API or an XML for a custom API, which allows the generation of Interface Field Mapping Template documents for custom APIs created during implementation.

The tool generates the Interface Field Mapping Template document as a Microsoft Excel XML spreadsheet document that can be opened in Microsoft Excel and modified to specify the mapping details.

Generating Interface Field Mapping Template Documents Using the Generation Tool

About this task

To generate the XML spreadsheet:

Procedure

Use the following command line tool:

```
generateExcelXML {INXML} {INXSL} {OUTXML} {HTML} {TITLE}
```

Results

where,

- INXML – Name of the XML file for which the XML spreadsheet should be generated
- INXSL – Name of the XSL file which is used to generate the XML spreadsheet
- OUTXML – Name of the XML spreadsheet file to be generated
- HTML – Name of the HTML file which contains the description of the Input XML attributes.

Note: If you are running the RDT in a Unix environment, you must insert an extra "\" for every "\" that you use in the HTML file name attribute. For example, if the filename is \\server\directory\file.html, you must specify the filename as \\server\directory\file.html.

- TITLE - The title that is displayed after you generate the XML spreadsheet.

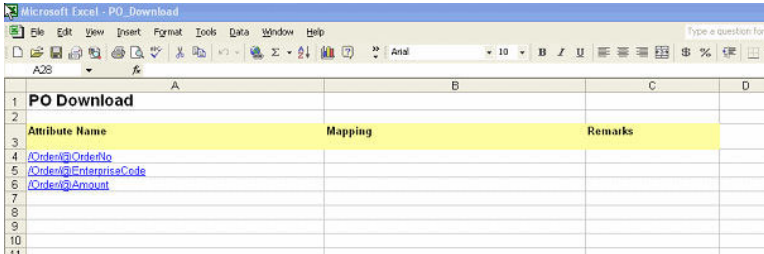
This tool is located in <INSTALL_DIR>/bin directory. This can also be used to generate XML spreadsheets for custom APIs.

Using Interface Field Mapping Template Documents

About this task

The XML spreadsheet generated using the command line tool can be opened and edited using Microsoft Excel (Versions 2002 and above).

The XML spreadsheet provides the Attribute Name, Mapping, and Remarks for each attribute. The following is a sample:



Attribute Name	Mapping	Remarks
/Order/@OrderNo		
/Order/@EnterpriseCode		
/Order/@Amount		

Procedure

1. Click on an attribute name to launch the relevant datatype and description.
2. Modify the integration field mappings as applicable and save.

Initial Data Loading

Sterling Selling and Fulfillment Foundation provides an initial data-loading tool for loading configuration data from legacy or ERP systems. The Initial Data Loading (IDL) tool utilizes the bare minimum information required by the warehouse to be functional.

Initial Data-Loading Services

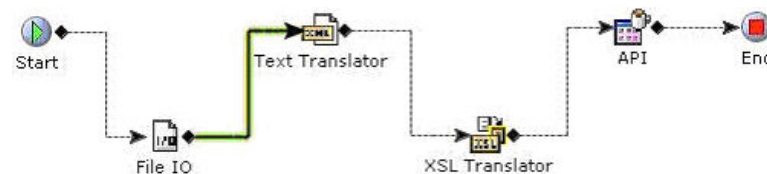
The Initial Data Loading (IDL) tool works based on the Service Definition Framework (SDF).

The IDL tool provides services to create the following configuration data:

- Items
- Shipping Cartons
- Locations
- SKU Dedications
- Location Inventory

To use the services provided for IDL, the configuration data to be loaded from the legacy or ERP systems should be made available in a comma delimited flat file.

The IDL tool uses services to convert the data into the XML format, required by the corresponding APIs to create or modify the relevant information in the warehouse. The following figure illustrates a sample service as displayed in the Applications Manager:



To begin the initial data loading process, the integration server should be started by navigating to the `<runtime>/bin` folder and entering the following command:

```
<runtime>/bin/startIntegrationServer.sh <servername>
```

For more information about running the Sterling Selling and Fulfillment Foundation Integration Server, refer to the *Sterling Selling and Fulfillment Foundation: Installation Guide*.

The `RDTConfigDataFormat.xls` file located in the `<runtime>/repository/xapi/template/merged/RDTConfigSchemas` folder contains the data sequence and the headers required for the corresponding service provided in the IDL module of the RDT.

All Sterling Selling and Fulfillment Foundation services follow the predefined sequence specified in the `RDTConfigDataFormat.xls` file for calling the components:

- The File IO Receiver is used to read the data from the delimited flat file
- The Text Translator component is used to convert the delimited data to XML format
- The XSL Translator component is used to convert the XML into a format that is the input to an API, and

- The API component is used to call the business API for creating or modifying the data.

Each service reads the input data line by line from the delimited flat files. Thus, all the details required for a configuration should be provided in a single line, separated by commas, and in a fixed sequence. The first item in each line is the header, and it is fixed for each service. If the first item is anything other than the header then that row is not considered for processing.

Error Handling in Initial Data Loading (IDL) Tool

The error handling for Initial Data Loading services is undertaken at two levels:

1. When there is an error in translating the flat file into an xml file as per the defined schema, the file is pushed to the working directory and an error file indicating the error is added to the error directory. The error may now be fixed and the modified flat file reprocessed.
2. When the API throws an exception for a record, it is sent to the default exception queue where it can be viewed in the exception console by searching for exceptions in initial status. The input xml may now be modified by providing the right input, and reprocessed using the reprocess button.

Item Configuration Data-Loading

This service enables you to create an item or modify the attributes of an existing item for which inventory is stored in the warehouse. It calls the `manageItem()` API.

The following table explains the format of the headers and the sequence of items to be provided for this service. For more information, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Attribute	Description	Sequence	Data Type	Size
ITEMHEDR	The item header identifier	1	String	6
ItemID	The unique identifier for an item that belongs to a catalog organization	2	String	40
OrganizationCode	The code of the organization whose product information is being stored	3	String	24
UnitOfMeasure	The unit of measure for item quantity	4	String	40
GlobalItemID	The unique global identifier used to cross reference an item with another catalog organization	5	String	128
Description	A localized description	6	String	200
ProductLine	The product line of an item	7	String	100
KitCode	The kit code of an item. Value 'LK' indicates a logical kit, while PK indicates a physical kit	8		
ItemGroupCode	The code of the item group. This is used to identify whether the item is a Product, Provided Service, Provided Service Option, Delivery Service, or Delivery Service Option	9	String	20

Attribute	Description	Sequence	Data Type	Size
UnitCost	The cost of one unit of the item	10	Decimal	19
CostCurrency	The currency in which the item's cost is defined	11	String	20
CountryOfOrigin	The item's country of origin or manufacture	12	String	40
ItemType	The generic type of the item	13	String	40
UnitWeight	The weight of one unit of the item	14	Decimal	14
WeightUOM	The unit of measure in which the item's weight is defined	15	Decimal	14
UnitHeight	The height of one unit of the item	16	Decimal	14
UnitLength	The length of one unit of the item	17	Decimal	14
UnitWidth	The width of one unit of the item	18	Decimal	14
SerializedFlag	This indicates whether the item is serialized	19	Boolean	1
TagControlFlag	This indicates whether the item is tag controlled	20	Boolean	1
TimeSensitive	This indicates whether the item is time sensitive	21	Boolean	1
IsFifoTracked	This indicates whether the item is FIFO tracked	22	Boolean	1
IsSerialTracked	This indicates whether the item is serial tracked	23	Boolean	1
HarmonizedCode	The harmonized code of the item	24	String	40
NMFCCode	The NMFC code of the item	25	String	40
VelocityCode	The velocity code of the item	26	String	40
ECCNNo	The ECCN number of the item	27	String	40
HazmatClass	The hazardous material classification of the item	28	String	40
CommodityCode	The commodity code of the item	29	String	40
StorageType	The storage type of the item	30	String	40
AddName1	The name of the first additional attribute	31	String	20
AddValue1	The value of the first additional attribute	32	String	2000
AddName2	The name of the second additional attribute	33	String	20
AddValue2	The value of the second additional attribute	34	String	2000
LotNumber	The lot number of the item. This indicates whether this attribute can be used as a Tag Identifier or a Tag Descriptor. Valid values are: 01 - Use as Tag Descriptor 02 - Use as Tag Identifier 03 - Not used	35	String	2

Attribute	Description	Sequence	Data Type	Size
LotAttribute1	The lot attribute of the item. This indicates whether this attribute can be used as a Tag Descriptor. Valid values are: 01 - Use as Tag Descriptor 03 - Not Use	36	String	2
LotAttribute2	The lot attribute of the item. This indicates whether this attribute can be used as a Tag Descriptor. Valid values are: 01 - Use as Tag Descriptor, 03 - Not used.	37	String	2
CaseQuantity	The quantity of one case of the item	38	Decimal	14
CaseWeight	The weight of one case of the item	39	Decimal	14
CaseLength	The length of one case of the item	40	Decimal	14
CaseWidth	The width of one case of the item	41	Decimal	14
CaseHeight	The height of one case of the item	42	Decimal	14
PalletQuantity	The quantity of one pallet of the item	43	Decimal	14
PalletWeight	The weight of one pallet of the item	44	Decimal	14
PalletLength	The length of one pallet of the item	45	Decimal	14
PalletWidth	The width of one pallet of the item	46	Decimal	14
PalletHeight	The height of one pallet of the item	47	Decimal	14
DimensionUOM	The unit of measure that define the dimensions of the item	48	String	40

Schema Files Used

The schema files used by each component of the service and the API called by the service are as follows:

- **Service Name:** Items
- **Service Group:** InitialDataLoad
- **Text Translator:** ModifyItemSchema
- **XSL Translator:** ModifyItem
- **API:** manageItem
- **Server Name:** ItemLoader

Shipping Carton Data-Loading

This service creates shipping cartons (modelled as items) that are stored in the warehouse. It calls the createItem() API.

The following table explains the format of the headers and the sequence of items to be provided for this service. For more information, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Attribute	Description	Sequence	Data Type	Size
ITEMHEDR	The item header identifier	1	String	6
ItemID	The unique identifier for an item that belongs to a catalog organization	2	String	40
OrganizationCode	The code of the organization whose product information is being stored	3	String	24
UnitOfMeasure	The unit of measure for item quantity	4	String	40
UnitWeight	The weight of one unit of the item	5	Decimal	14
UnitHeight	The height of one unit of the item	6	Decimal	14
UnitLength	The length of one unit of the item	7	Decimal	14
UnitWidth	The width of one unit of the item	8	Decimal	14
MaxCntrWeight	The maximum weight of the carton	9		

Schema Files Used

The schema files used by each component of the service and the API called by the service are as follows:

- **Service Name:** ShippingCartons
- **Service Group:** InitialDataLoad
- **Text Translator:** ShippingCartonSchema
- **XSL Translator:** ShippingCarton
- **API:** createItem
- **Server Name:** ShippingCartonLoader

Location Data-Loading

This service creates locations in a zone within a node in the warehouse. These locations specify the physical space where inventory is stored. It calls the `manageLocation()` API.

The following table explains the format of the headers and the sequence of items to be provided for this service. For more information, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Attribute	Description	Sequence	Data Type	Size
LOCAHEDR	The location header identifier	1	String	8
LocationId	The unique identifier for the location. This in conjunction with NODE_KEY identifies a unique location in the node	2	String	40
Node	The node to which the location belongs.	3	Key	24

Attribute	Description	Sequence	Data Type	Size
LocationType	The system defined classification of location to aid association of locations of certain types for certain other operations with WMS. The supported types are: INTRANSIT (Mobile locations), STAGING, VIRTUAL, REGULAR and DOCK. For example, all equipment locations should be of type INTRANSIT. If LocationType is passed blank or passed unallowed values then default LocationType is taken as REGULAR	4	String	40
ZoneId	The zone to which the location belongs. This in conjunction with the node key identifies a unique zone within the node.	5	String	40
AisleNumber	The aisle number of the location. Locations belong to zones, which have travel aisle's between them. A zone could belong to multiple aisles and multiple zones could belong to an aisle. But a location in a zone belongs to one and only one aisle.	6	Integer	9
LevelLocation	The level number of the location. This indicates the height of the location (y-co-ordinate of the location from the floor) classified as levels. Level attribute of the location is used in arriving at locations nearest to the dedicated locations algorithm used in put away. Typically, the level attribute is contained within the location ID.	7	Integer	9
BayNumber	The bay number of the location. Typically, the aisle, level and bay put together gives the physical location of the location in the node if they are based on coordinate system. Bay attribute of the location (x-coordinate from the beginning of the aisle) is used in arriving at locations nearest to the dedicated locations algorithm used in put away. Typically, the bay attribute is contained within the location ID.	8	Integer	9

Attribute	Description	Sequence	Data Type	Size
MoveInSequenceNumber	The move in sequence number of the location. This is used by task management for location suggestion while moving in inventory (put away). The put away location selection algorithm uses this information to select locations amongst a list of locations based on its move in sequence.	9	Integer	9
MoveOutSequenceNumber	The move out sequence number of the location. This is used by task management for location suggestion while moving out inventory (picking). The pick location selection algorithm uses this information to select locations amongst a list of locations based on its move in sequence.	10	Integer	9
InStagingLocationId	The in staging location id indicates the Drop off location (For moves coming into a location, they may be dropped here)	11	String	40
OutStagingLocationId	The out staging location id indicates the Out Drop off Location (Location where moves originated at this location, may be dropped).	12	String	40

Attribute	Description	Sequence	Data Type	Size
VelocityCode	The velocity code of the location classifies items as A, B or C class items based on whether they are fast selling, not so fast selling and low selling item. These item classifications are typically followed by all enterprises to optimize certain operations such as sourcing and stocking. The reason we have locations preferring certain velocity codes is that, we could have locations closer to dock stocking A class items, and locations furthest away from the dock stocking C class items. Velocity code is a preference on the location and not a constraint. If A class items fill up all locations meant for A class items, then they can go in to B and then C. Similarly C can go to B and then A for lack of space in the respective locations preferred for a specific velocity code. B class items go into C and then into A. If VelocityCode is passed blank or passed unallowed values then default VelocityCode is taken Last VelocityCode in the alphabetic sequence in common code of type VELOCITY_CODE.	13	String	40
LocationSize Code	The location size code defines the capacity of a location. All locations having the same size (dimensions and ability to hold the same weight) are classified under the same size code. This maps to the primary key attribute of the YFS_LOCATION_SIZE_CODE table.	14	String	40
StorageCode	Storage code is an attribute of the location that allows the warehouse to store items that have the same storage profile as that of the location. For example, hazardous inflammable items need locations close to fire extinguishers. In this case the locations are marked as having a storage code, which is suitable for storing Inflammable items. This ensures that only inflammable items get to these locations.	15	String	40
X Co-ordinate	X Co-ordinate for a location in the warehouse	16	Number	14
Y Co-ordinate	Y Co-ordinate for a location in the warehouse	17	Number	14

Attribute	Description	Sequence	Data Type	Size
Z Co-ordinate	Z Co-ordinate for a location in the warehouse	18	Number	14

Schema Files Used

The schema files used by each component of the service and the API called by the service are as follows:

- **Service Name:** Locations
- **Service Group:** InitialDataLoad
- **Text Translator:** LocationSchema
- **XSL Translator:** Location
- **API:** manageLocation
- **Server Name:** LocationLoader

SKU Dedication Data-Loading

This service modifies the attributes of a location, and is basically used to dedicate a location as a dedicated location. A dedicated location is one that stores inventory for a particular item only. It calls the `modifyLocation()` API.

Note: This service require 9 attributes. If you are giving 8 commas to separate these 9 attributes, you have to make sure that the last attribute is non-blank. If it is blank, you have to close it with an extra comma, which means the 9th comma. In this case, 9 commas does not mean that there are 10 attributes.

The following table explains the format of the headers and the sequence of items to be provided for this service. For more information, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Attribute	Description	Sequence	Data Type	Size
SKUDEDIC	The SKU Dedication header identifier	1	String	8
LocationId	The identifier for the location. This in conjunction with NODE_KEY identifies a unique location in the node	2	String	40
Node	The node to which the location belongs to	3	Key	24
EnterpriseCode	The code of the enterprise to which the location is dedicated	4	String	40
ItemId	The item identifier of the SKU	5	String	40
UnitOfMeasure	The unit of measure of the SKU	6	String	40
ProductClass	The product class of the SKU	7	String	40
SegmentType	SKUs are sometimes custom made. This field stores the customization details.	8	String	40

Attribute	Description	Sequence	Data Type	Size
Segment	SKUs are sometimes custom made. This field stores the customization details. When inventory is customized for a specific order, it needs to be tracked separately so that it can be allocated to that order	9	String	40

Schema Files Used

The schema files used by each component of the service and the API called by the service are as follows:

- **Service Name:** SkuDedications
- **Service Group:** InitialDataLoad
- **Text Translator:** SkuDedicationSchema
- **XSL Translator:** SkuDedication
- **API:** modifyLocation
- **Server Name:** SkuDedicationLoader

Location Inventory Data-Loading

This service adds the inventory for the previously created items and locations in the warehouse. It calls the `adjustLocationInventory()` API.

The following table explains the format of the headers and the sequence of items to be provided for this service. For more information, see the *Sterling Selling and Fulfillment Foundation: Javadocs*.

Table 16. Format of Headers and Sequence of Items for this Service

Attribute	Description	Sequence	Data Type	Size
ALOCINVN	The inventory header identifier	1	String	8
EnterpriseCode	The Inventory Organization Code. This indicates the Enterprise whose product information is being stored.	2	String	24
Node	The Business key or unique identifier for a ship node.	3	String	24
CaseId	The identifier for a case. This gives the LPN information for adjustment.	4	String	40
LocationId	The identifier for a location. This forms unique key of this table in conjunction with NODE_KEY. Indicates the location from where the inventory is being adjusted. LocationId becomes mandatory, if CaseId/PalletId is not passed.	5	String	40
PalletId	The identifier for a pallet. This gives the LPN information for adjustment.	6	String	40

Table 16. Format of Headers and Sequence of Items for this Service (continued)

Attribute	Description	Sequence	Data Type	Size
InventoryStatus	The inventory status gives the status of the inventory. Only one level InventoryStatus transitions happen for the inventory for positive adjustments. Negative adjustments do not take care of InventoryStatus transitions. If not passed, the status is taken as blank.	7	String	10
SegmentType	The segment type for particular enterprise or organization. SegmentType becomes mandatory if Segment is passed.	8	String	40
Segment	The segment for particular enterprise or organization. Segment becomes mandatory if SegmentType is passed.	9	String	40
Quantity	This gives the adjustment quantity for the inventory. The negative quantity specifies negative adjustment and positive quantity denotes positive adjustment. Quantity becomes mandatory if SerialDetail does not provide quantity for adjustment.	10	Decimal	14
ItemID	The item identity for the inventory	11	String	40
UnitOfMeasure	The unit of measure for the item	12	String	40
ProductClass	The product class for the item	13	String	40
LotNumber	The lot number for the inventory	14	String	40
LotAttribute1	The lot attribute for the inventory	15	String	40
LotAttribute2	The lot attribute for the inventory	16	String	40
ShipByDate	The date by which the inventory has to be shipped	17	Date	10
SerialNo	The unique identifier for each serial	18	String	40

Table 16. Format of Headers and Sequence of Items for this Service (continued)

Attribute	Description	Sequence	Data Type	Size
ReasonCode	The reason code for the inventory transaction. The business significance of this reason code is that inventory bins are tied to this reason code, which is used to adjust inventory (for global inventory visibility purposes) on host systems. This is mandatory if inventory is getting updated. Some Sterling Selling and Fulfillment Foundation APIs doing inventory adjustments expect some adjustment reason codes to be configured in the system. These are RECEIPT used by Receiving, PACK used by Packing functions and SHIP used by Shipment. PACK should have a bin associated while RECEIPT and SHIP should not have bin location associations.	19	String	40

Schema Files Used

The schema files used by each component of the service and the API called by the service are as follows:

- **Service Name:** Inventory
- **Service Group:** InitialDataLoad
- **Text Translator:** AdjustLocationInventorySchema
- **XSL Translator:** AdjustLocationInventory
- **API:** adjustLocationInventory
- **Server Name:** InventoryLoader

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2011. Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2011.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Connect Control Center[®], Connect:Direct[®], Connect:Enterprise[™], Gentran[®], Gentran[®]:Basic[®], Gentran:Control[®], Gentran:Director[®], Gentran:Plus[®], Gentran:Realtime[®], Gentran:Server[®], Gentran:Viewpoint[®], Sterling Commerce[™], Sterling Information Broker[®], and Sterling Integrator[®] are trademarks or registered trademarks of Sterling Commerce[™], Inc., an IBM Company.

Other company, product, and service names may be trademarks or service marks of others.



Printed in USA